AN EFFICIENT NUMERICAL TECHNIQUE

FOR

CALCULATING THERMAL SPREADING RESISTANCE

Final Report

8 February 1973

(NASA-CR-12412) AN EFFICIENT NUMERICAL TECHNIQUE FOR CALCULATING THERMAL SPREADING RESISTANCE Final Report (General Electric Co.) 84 p HC $6.25 CSCL 20M G3/33

N73-28921

Unclas 15033

A Study Prepared for

National Aeronautics and Space Administration
under Contract NAS8-28516

Prepared by

Dr. Earl H. Gale, Jr.
General Electric Company
Aerospace Electronic Systems Department
Utica, New York

ACKNOWLEDGMENTS

The author wishes to express his appreciation to the following General Electric people for their support on this project:

Stephen A. Smith, who reviewed the basic technique and made calculations of the computer work required for this and several other techniques.

Edwin Kelly, who translated my program from BASIC to the FORTRAN Y version listed in the Appendix of this report.

Claude Lindeman, who contributed the section on the use of superposition.

AN EFFICIENT NUMERICAL TECHNIQUE

FOR

CALCULATING THERMAL SPREADING RESISTANCE

Final Report

8 February 1973

A Study Prepared for

National Aeronautics and Space Administration
under Contract NAS8-28516

Prepared by

Dr. Earl H. Gale, Jr.
General Electric Company
Aerospace Electronic Systems Department
Utica, New York

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

This final report has been prepared by the General Electric Company, Aerospace Electronic Systems Department, Utica, New York under contract NAS8-28516. The report documents the results of a thermal spreading resistance data generation technique study. The method developed is discussed in detail, illustrative examples given, and the resulting computer program is included.

SECTION II

BACKGROUND

A. GENERAL

"Thermal spreading resistance" is defined as the conductive thermal resistance between a source region and a sink region in a solid where the geometry is such as to preclude one dimensional heat flow.

Knowledge of thermal spreading resistance is needed in two aerospace engineering areas. These are the thermal design of electronic components or equipments and in the prediction and control of thermal contact resistance.

1. Importance To The Design Of Electronic Components and Equipments

The thermal analysis of a power semiconductor or integrated circuit can be reduced to the problem of determining the appropriate spreading and bonding thermal resistances. As an example, the problem of calculating the junction-to-case thermal resistance of a semiconductor bonded to a substrate which is bonded in a metal case will be considered. Figure 1 illustrates this problem.

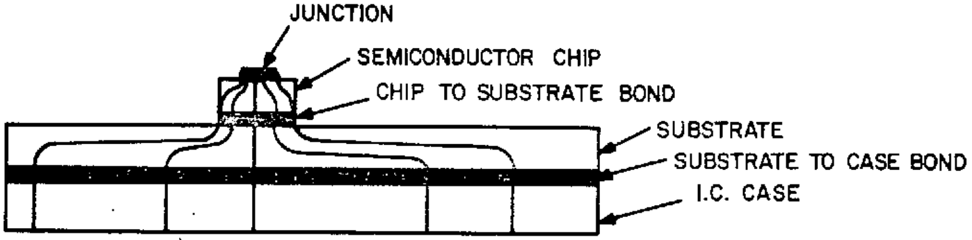


Figure 1. Semiconductor in an Integrated Circuit

Heat is generated in a region of known size, the junction region of the semiconductor. The first, and most significant, spreading resistance of interest occurs between the junction and the opposite face of the silicon chip. The next thermal resistance of interest is that across the bond between chip and substrate. It is of significance that these thermal resistances are not independent although many thermal designers, under the pressures of a design schedule, have treated them as such. The thermal conductance of the bond proper can vary several thousandfold depending on the use of a metallic or nonmetallic bonding material. The resistance to heat flow between the semiconductor chip bond region and the rear of the substrate represents a second spreading resistance, etc. In a typical integrated circuit package the entire bottom region of the substrate would not be available as a sink for a single semiconductor chip due to the presence of other heat dissipating chips. It is usually possible to estimate the effective sink region on the rear of the substrate from considerations of symmetry or because it exceeds dimensions which appreciably affect the thermal spreading resistance. In those few cases where interactions must be considered, the key analytical tool is superposition; Green's function approach

may also be employed to advantage. For example, see reference 1 and the discussion beginning on page 37 of this report.

The importance of being able to predict thermal spreading resistances in single and multi- layered material in the evaluation of the thermal design of semiconductor or integrated circuits has been shown. Spreading thermal resistances are important in other electrical devices such as phased array antenna elements, Peltier coolers, Seebeck generators and many devices which utilize conductive heat transfer.

B. PREDICTION AND CONTROL OF THERMAL CONTACT RESISTANCE

The resistance to heat flow between two mating (touching, as in a joint) pieces of metal is called thermal contact resistance. When the actual microscopic regions of contact between two mating surfaces are examined, it is found that metal-to-metal contact occurs in small discrete regions where the asperities or microscopic protuberances make contact. References 2 and 3 describe this model of contact in great detail. Figure 2 illustrates this contact model.

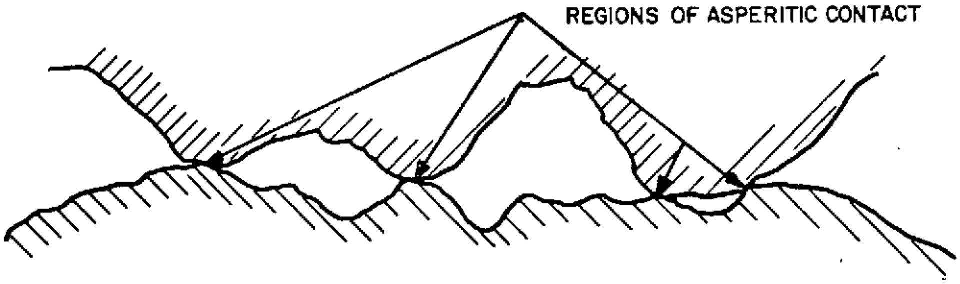


Figure 2. Microscopic View of the Joint of Contacting Pieces of Metal

The heat flow to and from a region of asperitic contact into the contacting proper is seen to be of the "spreading" type. In fact, the effective thermal contact resistance of any contact may be considered as the sum of the parallel microscopic spreading resistances in the contacts themselves. References 2 and 3 above deal largely with isentropic contacts in which the thermal conductivity within the bodies of both contacts is uniform.

Analysis has shown that the bulk of the spreading resistance occurs close to the region of actual asperitic contact and that the spreading resistance in any region varies inversely with the thermal conductivity of the material. Figures 3 and 4 illustrate the first of these points. Figure 3, drawn to scale, shows the equipotential lines about a circular contact region each drawn to show one-tenth of the total spreading resistance between the circular source region and the body of a very large contact. It is seen that half of this total resistance occurs within one contact radius from the circular contact or source region and 80 percent occurs within three contact radii. Figure 4 illustrates these relationships. Figures 3 and 4 are taken from reference 4.

The thermal conductivity of contact close to the surface is of such importance that even a thin 45 Angstrom thick layer of oxide on an aluminum contact can contribute measurably to the thermal contact resistance of an aluminum contact. This has been shown by Gale, reference 4.

Mikic and Carnasciali, reference 5, have utilized the above principle to enhance thermal contact conductance by plating materials of higher conductivity on the contacting faces of a

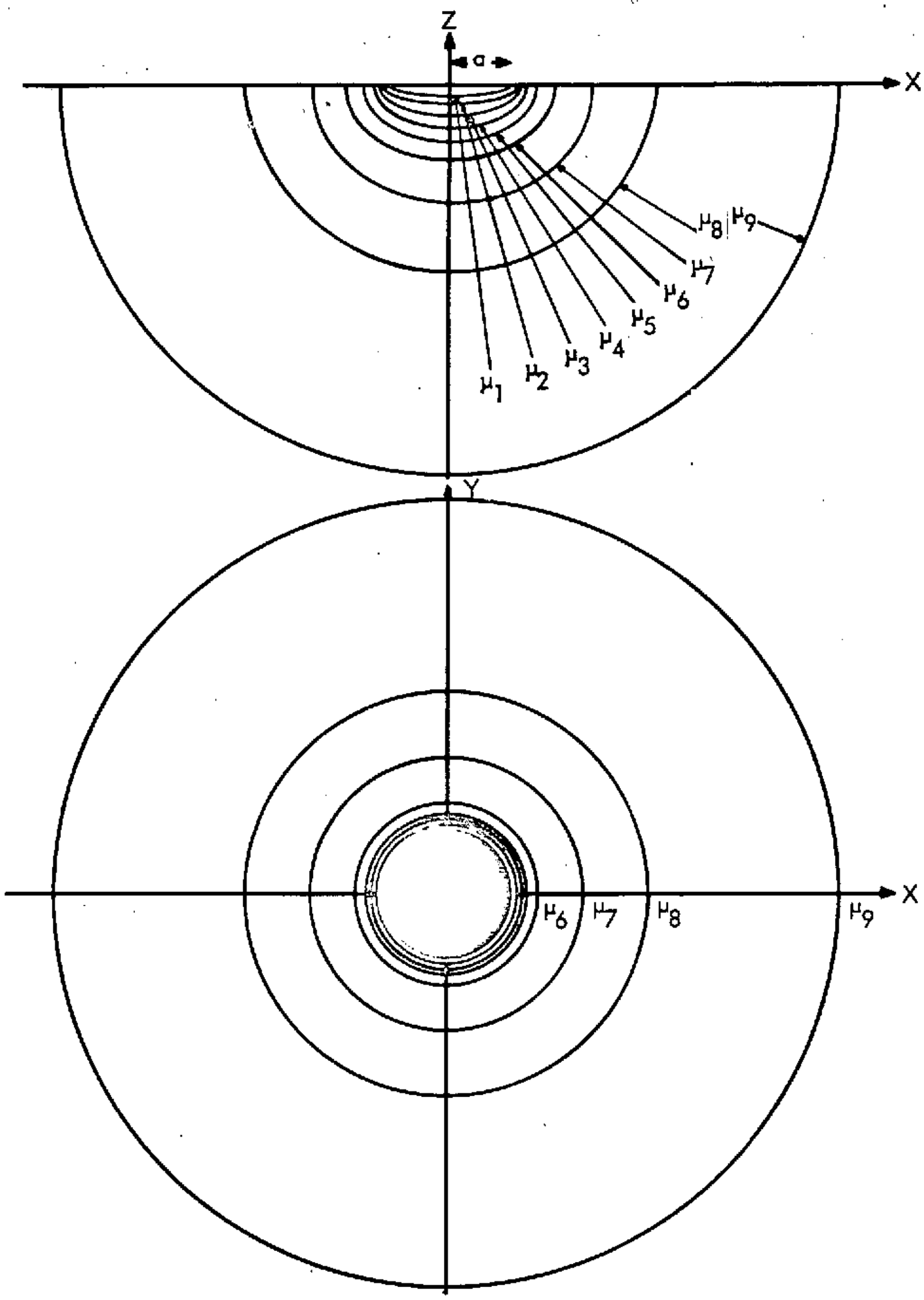


Figure 3. Temperature Profiles Described by Holm's Equation for Isothermal Circular Source on a Semi-infinite Slab

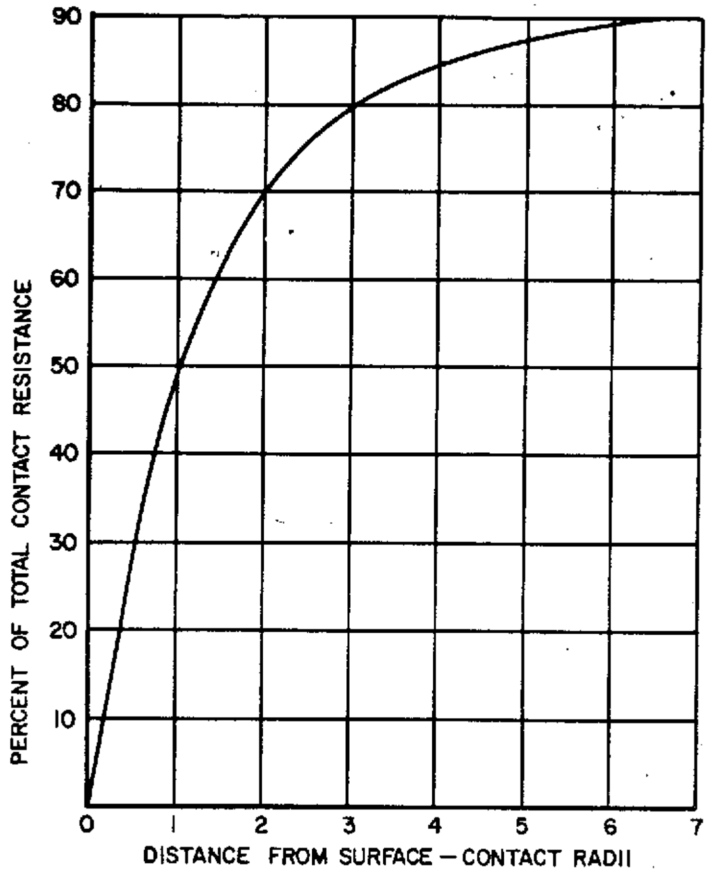


Figure 4. Percent of Total Constriction Resistance for a Single Isothermal Circular Source on a Semi-infinite Slab as a Function of Distance into Body of Contact

metallic joint. They have attempted an analysis of spreading resistance from a circular contact into a contact composed of two layers of materials with different conductivities. An exact boundary value solution of this basic problem has proven too difficult as no mathematical function has been found which will satisfy the boundary conditions between the plating and the body materials.

Professor C. J. Moore, Jr.[1] in his discussion printed at the end of reference 5 felt this two layered spreading resistance problem could best be handled by a "well-conditioned finite difference computer code." Mikic and Carnasciali then question the economic feasibility of such calculations.

Attempts by the author of this study to solve the two layer thermal spreading resistance problem using a finite difference approach utilizing Gauss-Seidel iteration have shown the cost of digital computer calculation to be great.

SECTION III

THEORY

A. GENERAL

The governing differential equation for the thermal spreading resistance problem is Poisson's equation. For those spreading resistance problems that are two-dimensional[2] or may be reduced to two-dimensional problems, the equation is:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = q_t''' \tag{1}$$

Consider a rectangular field subdivided into rectangular subregions as illustrated in Figure 5. The heat balance equation describing the heat flow among element m, n and its four principal neighbors is:

$$(T_{m,n} - T_{m,n+1}) H_{m,n} + (T_{m,n} - T_{m-1,n}) V_{m,n} + (T_{m,n} - T_{m,n-1}) H_{m,n-1}$$

$$+ (T_{m,n} - T_{m+1,n}) V_{m+1,n} = q_{m,n} \tag{2}$$

where:

- T is temperature
- q''' heat generated per unit volume
- x, y, z are spatial coordinates
- H, V are horizontal and vertical conductances, respectively
- $q_{m,n}$ heat generated in mode m, n

The convention for the horizontal and vertical conductances used is shown in Figure 6.

Each of the following observations below will be helpful in understanding the discussion which follows:

(1) When any temperature $t_{m,n}$ is known (e.g., as a boundary condition), it will affect equation m, n by yielding a term $q_{m,n}$ which is subtracted from the right hand side of equation (2) where $q_{m,n}'$ is:

$$q_{m,n}' = T_{m,n}(H_{m,n} + V_{m,n} + H_{m,n-1} + V_{m+1,n}) \tag{3}$$

---

[1] Associate Professor of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, N.C.

[2] The method developed is applicable to three-dimensional problems as will be shown later in the report.

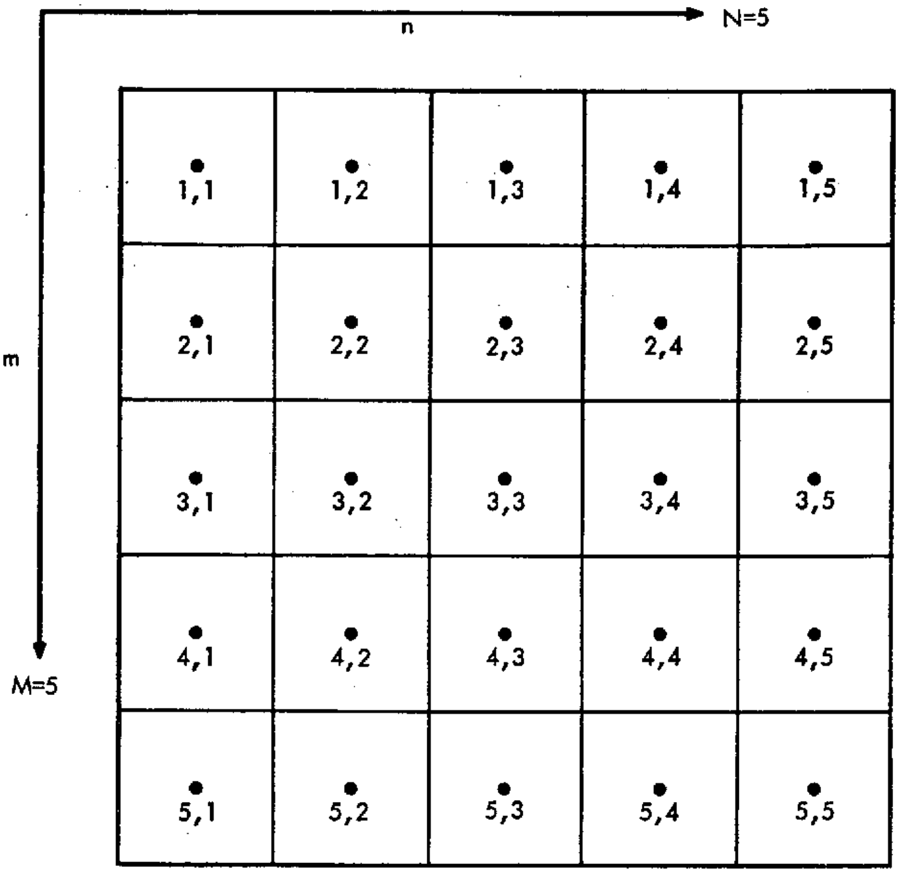
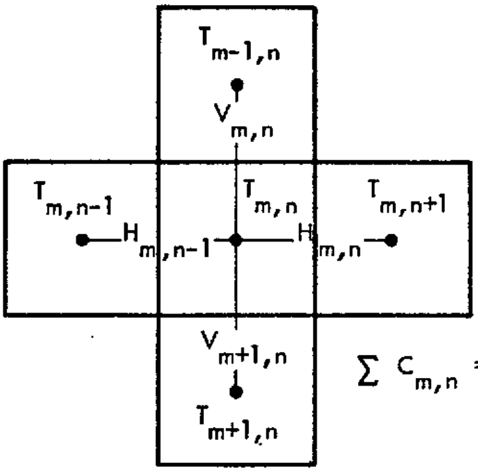


Figure 5. Rectangular Field Divided into 25 Finite Elements



$$\sum C_{m,n} = H_{m,n} + V_{m+1,n} + H_{m,n-1} + V_{m,n}$$

Figure 6. Nomenclature for Nodal Interconductances

(2) If the original field is divided into M rows and N columns, and further if M = N, then:

(a) There will be N^2 linear equations.

(b) There will be not more than N^2 unknowns (fewer if some temperatures are initially prescribed).

(c) There can be as many different and distinct nodal conductances as there are interconnections between nodes.

Now, if the system of linear finite difference equations is written in matrix form (taking the nodes of Figure 5 into consideration) from left to right, top row to bottom row, as in reading English, a coefficient matrix results that has a pattern characteristic for field problems described by Poisson's or LaPlace's equations. This pattern is illustrated in Figure 7.

It was noted by Karlqvist (Reference 6) that the matrix in Figure 7 may be partitioned as shown. It can be seen that each of the submatrices is N x N and the coefficient matrix is $N^2 \times N^2$ where the original finite element matrix was N x N in size.

B. DERIVATION OF AN EFFICIENT TECHNIQUE FOR EXACT SOLUTION OF THIS SYSTEM OF EQUATIONS

Defining the sub-matrices shown in Figure 7 as follows:

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & 0 \\ 0 & 0 & A_4 & B_4 & C_4 \\ 0 & 0 & 0 & A_5 & B_5 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \end{bmatrix}$$

Figure 8. System Of Submatrices In Matrix Notation

Expanding the partitioned matrices (Figure 8) into a system of equations, having normalized each equation with respect to the diagonal element:

$$\begin{array}{ccccccc}
T_1 & B_1^{-1}C_1T_2 & 0 & 0 & 0 & = & B_1^{-1}Q_1 \\
B_2^{-1}A_2T_1 & T_2 & B_2^{-1}C_2T_3 & 0 & 0 & = & B_2^{-1}Q_2 \\
0 & B_3^{-1}A_3T_2 & T_3 & B_3^{-1}C_3T_4 & 0 & = & B_3^{-1}Q_3 \\
0 & 0 & B_4^{-1}A_4T_3 & T_4 & B_4^{-1}C_4T_5 & = & B_4^{-1}Q4 \\
0 & 0 & 0 & B_5^{-1}A_5T_4 & T_5 & = & B_5^{-1}Q_5
\end{array}$$

Upon redefining constants in the following manner:

$B_2^{-1}A_2 \equiv -B_2$, $B_2^{-1}C_2 \equiv -A_2$, and $B_2^{-1}Q_2 \equiv C_2$, etc.

$$
\left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc}
\Sigma C_{1,1} & -H_{1,1} & 0 & 0 & 0 & -V_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-H_{1,1} & \Sigma C_{1,2} & -H_{1,2} & 0 & 0 & 0 & -V_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -H_{1,2} & \Sigma C_{1,3} & -H_{1,3} & 0 & 0 & 0 & -V_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -H_{1,3} & \Sigma C_{1,4} & -H_{1,4} & 0 & 0 & 0 & -V_{2,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -H_{1,4} & \Sigma C_{1,5} & 0 & 0 & 0 & 0 & -V_{2,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-V_{2,1} & 0 & 0 & 0 & 0 & \Sigma C_{2,1} & -H_{2,1} & 0 & 0 & 0 & -V_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -V_{2,2} & 0 & 0 & 0 & -H_{2,1} & \Sigma C_{2,2} & -H_{2,2} & 0 & 0 & 0 & -V_{3,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -V_{2,3} & 0 & 0 & 0 & -H_{2,2} & \Sigma C_{2,3} & -H_{2,3} & 0 & 0 & 0 & -V_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -V_{2,4} & 0 & 0 & 0 & -H_{2,3} & \Sigma C_{2,4} & -H_{2,4} & 0 & 0 & 0 & -V_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -V_{2,5} & 0 & 0 & 0 & -H_{2,4} & \Sigma C_{2,5} & 0 & 0 & 0 & 0 & -V_{3,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & -V_{3,1} & 0 & 0 & 0 & 0 & \Sigma C_{3,1} & -H_{3,1} & 0 & 0 & 0 & -V_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -V_{3,2} & 0 & 0 & 0 & -H_{3,1} & \Sigma C_{3,2} & -H_{3,2} & 0 & 0 & 0 & -V_{4,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{3,3} & 0 & 0 & 0 & -H_{3,2} & \Sigma C_{3,3} & -H_{3,3} & 0 & 0 & 0 & -V_{4,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{3,4} & 0 & 0 & 0 & -H_{3,3} & \Sigma C_{3,4} & -H_{3,4} & 0 & 0 & 0 & -V_{4,4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{3,5} & 0 & 0 & 0 & -H_{3,4} & \Sigma C_{3,5} & 0 & 0 & 0 & 0 & -V_{4,5} & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{4,1} & 0 & 0 & 0 & 0 & \Sigma C_{4,1} & -H_{4,1} & 0 & 0 & 0 & -V_{5,1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{4,2} & 0 & 0 & 0 & -H_{4,1} & \Sigma C_{4,2} & -H_{4,2} & 0 & 0 & 0 & -V_{5,2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{4,3} & 0 & 0 & 0 & -H_{4,2} & \Sigma C_{4,3} & -H_{4,3} & 0 & 0 & 0 & -V_{5,3} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{4,4} & 0 & 0 & 0 & -H_{4,3} & \Sigma C_{4,4} & -H_{4,4} & 0 & 0 & 0 & -V_{5,4} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{4,5} & 0 & 0 & 0 & -H_{4,4} & \Sigma C_{4,5} & 0 & 0 & 0 & 0 & -V_{5,5} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{5,1} & 0 & 0 & 0 & 0 & \Sigma C_{5,1} & -H_{5,1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{5,2} & 0 & 0 & 0 & -H_{5,1} & \Sigma C_{5,2} & -H_{5,2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{5,3} & 0 & 0 & 0 & -H_{5,2} & \Sigma C_{5,3} & -H_{5,3} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{5,4} & 0 & 0 & 0 & -H_{5,3} & \Sigma C_{5,4} & -H_{5,4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V_{5,5} & 0 & 0 & 0 & -H_{5,4} & \Sigma C_{5,5}
\end{array}\right]
\left[\begin{array}{c}
T_{1,1} \\ T_{1,2} \\ T_{1,3} \\ T_{1,4} \\ T_{1,5} \\ \hline
T_{2,1} \\ T_{2,2} \\ T_{2,3} \\ T_{2,4} \\ T_{2,5} \\ \hline
T_{3,1} \\ T_{3,2} \\ T_{3,3} \\ T_{3,4} \\ T_{3,5} \\ \hline
T_{4,1} \\ T_{4,2} \\ T_{4,3} \\ T_{4,4} \\ T_{4,5} \\ \hline
T_{5,1} \\ T_{5,2} \\ T_{5,3} \\ T_{5,4} \\ T_{5,5}
\end{array}\right]
=
\left[\begin{array}{c}
Q_{1,1} \\ Q_{1,2} \\ Q_{1,3} \\ Q_{1,4} \\ Q_{1,5} \\ \hline
Q_{2,1} \\ Q_{2,2} \\ Q_{2,3} \\ Q_{2,4} \\ Q_{2,5} \\ \hline
Q_{3,1} \\ Q_{3,2} \\ Q_{3,3} \\ Q_{3,4} \\ Q_{3,5} \\ \hline
Q_{4,1} \\ Q_{4,2} \\ Q_{4,3} \\ Q_{4,4} \\ Q_{4,5} \\ \hline
Q_{5,1} \\ Q_{5,2} \\ Q_{5,3} \\ Q_{5,4} \\ Q_{5,5}
\end{array}\right]
$$

Figure 7. System of Equations in Matrix Notation

the general equation has the form:

$$-B_iT_{i-1} + T_i - A_iT_{i+1} = C_i \tag{4}$$

The first equation can be solved for T_1:

$$T_1 = C_1 + A_1T_2 \tag{5}$$

and the ith for T_i:

$$T_i = C_i + A_iT_{i+1} + B_iT_{i-1} \tag{6}$$

The goal is to find a recursion relationship built upon successive substitutions, which provides a solution for the ith unknown in terms of the (i+1)th. That is:

$$T_i = A_i' T_{i+1} + B_i' \tag{7}$$

Examining equation (5) for T_1 above, it can be seen that:

$$A_1' = A_1 \text{ and } B_1' = C_1$$

The equation for T_2 is:

$$T_2 = C_2 + A_2T_3 + B_2T_1 \tag{8}$$

which, when written in terms of the equation for T_1, becomes:

$$T_2 = \left[I - B_2A_1\right]^{-1} A_2T_3 + \left[I - B_2A_1'\right]^{-1} \left[C_2 + B_2B_1'\right] \tag{9}$$

The general coefficients found in this manner become

$$A_i' = \left[I - B_iA_{i-1}'\right]^{-1} A_i$$

and

$$B_i' = \left[I - B_iA_{i-1}'\right]^{-1} \left[B_iB_{i-1}' + C_i\right]$$

Therefore

$$T_i = A_i' T_{i+1} + B_i' \tag{10}$$

The temperature matrices (columns) are found starting at the Nth row by making

$$T_N = B_N' \tag{11}$$

$A_N = 0$ as a boundary condition results in modification of C_N above [see equation (3)].

The system of equations has been solved by operating on $3\sqrt{N} - 2$ submatrices, each of which is the square root of the size of the original N x N coefficient. $3\sqrt[2]{N}$ inversions of these

submatrices are required. The total number of multiplications (an indication of the effort) required during solution is:

$$\text{No. of Multiplications} = 3N^2 + N^{3/2} - N + N^{1/2} \tag{12}$$

This may be compared against other direct methods (see Ref. 7):

Method	Number of Multiplications Required during Solution
Gaussian Elimination	$\frac{1}{3} N^3 + N^2 - \frac{1}{3} N$
Jordan	$\frac{1}{2} N^3 + N^2 - \frac{1}{2} N$
Doolittle	$\frac{1}{3} N^3 + N^2 - \frac{1}{3} N$
Cholesky	$\frac{1}{6} N^3 + \frac{3}{2} N^2 + \frac{1}{3} N$

Cornock's method (Ref. 8),a triangulation type, also makes use of the characteristic pattern of submatrices which results during a finite difference solution for fields described by Poisson's equation. When the field properties are homogeneous and isentropic, Cornock's method is very powerful since only one of the above submatrices of order $\sqrt{N}$ need be inverted. However, for the general solution of the nonhomogeneous field, the number of multiplications required is

$$\frac{13}{2} N^2 - \frac{3}{2} N^{3/2} - 2N - 5 \tag{13}$$

A serious drawback to Cornock's method is that it does not lend itself to ready general programming for matrices of variable size as does the method described in this report.

That equation (12) is indicative of the computer effort required for solution has been substantiated in practice. Figure 9 shows the variation in cost realized in the solution of very large matrices using the method developed in this report. Also, a strong feature of this method is that very large systems of equations, e. g., 2500, can be conveniently handled in a direct solution.

The FORTRAN Y program contained in the Appendix was used on a Honeywell 630 computer in generating the dollar costs shown in Figure 9. Out-of-core storage of submatrices was utilized for very large systems.

C. APPLICABILITY OF TECHNIQUE TO THREE-DIMENSIONAL PROBLEMS

The technique discussed above is suited to the solution of field problems having three or more dimensions. Figure 10 illustrates the characteristic pattern of the coefficient matrix for a three dimensional finite element array. It is seen that the submatrices are $\sqrt[3]{N}$ in size. The same block tridiagonal pattern of submatrices is seen to occur as in the two-dimensional case so the derivation above for the technique of solution for two dimensional matrices is still applicable. Thus, although the BASIC and FORTRAN Y program presented later in this report are written for two-dimensional problems, little revision of these programs is required to handle three dimensional programs. Since the submatrices are $\sqrt[3]{N}$ rather than the $\sqrt[2]{N}$ in size, the technique is even more powerful for three dimensional problems. The number of multiplications

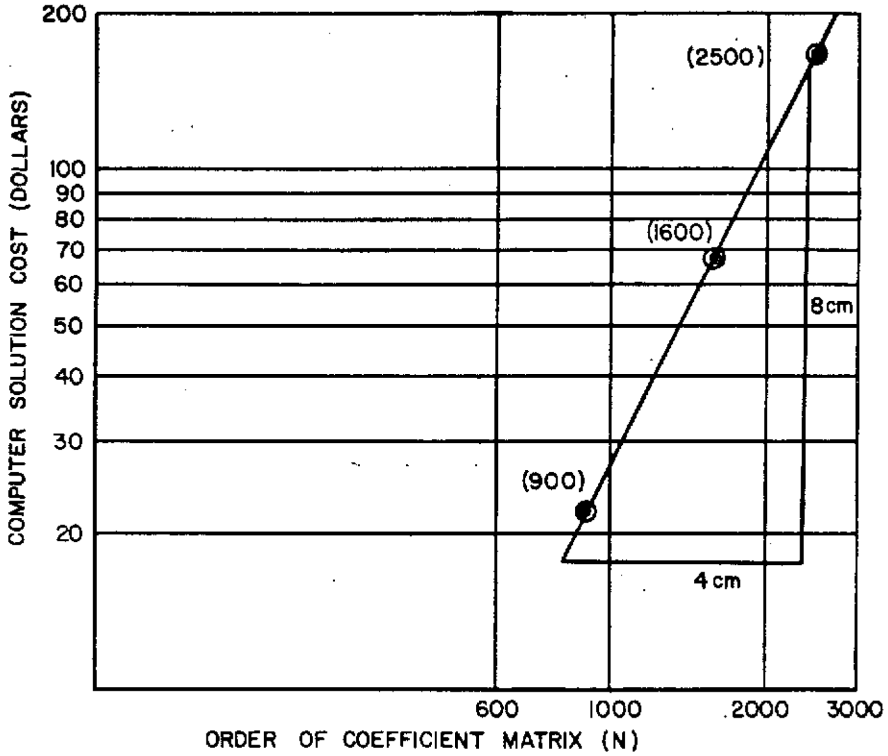
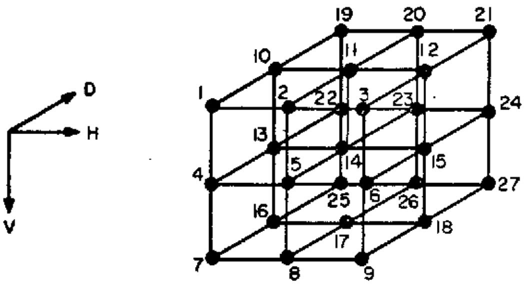


Figure 9. Cost of Computation vs Size of Coefficient Matrix

required for solution of the three-dimensional problem is a function of $N^{4/3}$ as opposed to N^2 for the two dimensional array where N is the order of the coefficient matrix.



N

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27			
1	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T_1		Q_1
2	H	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T_2		Q_2
3	0	H	Σ	0	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T_3		Q_3
4	V	0	0	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T_4		Q_4
5	0	V	0	H	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	0	T_5		Q_5
6	0	0	V	0	H	Σ	0	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	0	T_6		Q_6
7	0	0	0	V	0	0	Σ	H	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	0	T_7		Q_7
8	0	0	0	0	V	0	H	Σ	H	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	0	T_8		Q_8
9	0	0	0	0	0	V	0	H	Σ	0	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	0	0	T_9		Q_9
10	D	0	0	0	0	0	0	0	0	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	0	T_{10}		Q_{10}
11	0	D	0	0	0	0	0	0	0	H	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	0	0	T_{11}		Q_{11}
12	0	0	D	0	0	0	0	0	0	0	H	Σ	0	0	V	0	0	0	0	0	D	0	0	0	0	0	0	T_{12}		Q_{12}
13	0	0	0	D	0	0	0	0	0	V	0	0	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	0	T_{13}	=	Q_{13}
14	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	H	0	V	0	0	0	0	0	D	0	0	0	0	T_{14}		Q_{14}
15	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	0	0	V	0	0	0	0	0	D	0	0	0	T_{15}		Q_{15}
16	0	0	0	0	0	0	D	0	0	0	0	0	V	0	0	Σ	H	0	0	0	0	0	0	0	D	0	0	T_{16}		Q_{16}
17	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	H	0	0	0	0	0	0	0	D	0	T_{17}		Q_{17}
18	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	0	0	0	0	0	0	0	0	D	T_{18}		Q_{18}
19	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	0	Σ	H	0	V	0	0	0	0	0	T_{19}		Q_{19}
20	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	H	Σ	H	0	V	0	0	0	0	T_{20}		Q_{20}
21	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	0	0	H	Σ	0	0	V	0	0	0	T_{21}		Q_{21}
22	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	0	Σ	H	0	V	0	0	T_{22}		Q_{22}
23	0	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	H	0	V	0	T_{23}		Q_{23}
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	0	0	V	T_{24}		Q_{24}
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	0	Σ	H	0	T_{25}		Q_{25}
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	H	T_{26}		Q_{26}
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D	0	0	0	0	0	V	0	H	Σ	T_{27}		Q_{27}

WHERE $\Sigma_M = -\Sigma \; (V_m + H_m + D_m)$

Figure 10. Matrix Representation of System of Equations Describing Three-Dimensional Field Showing Pattern Established by Submatrices of Coefficient Matrix

SECTION IV

ILLUSTRATIVE PROBLEM AND PROGRAM

A sample thermal spreading resistance problem will be solved to illustrate the technique presented in Section II. The computer program used in the problem is written in BASIC language. A general version of the same program written in FORTRAN Y is included in the Appendix.

A. DESCRIPTION OF SAMPLE PROBLEM

The thermal spreading resistance problem to be considered is depicted in Figure 11. Heat is uniformly generated in a plane circular region of radius a and flows to a circular sink of radius b both concentric with, and parallel to, the source region a distance H away in a conductive medium. The conductive medium is divided into two regions of different conductivity. An exact closed form solution of this problem has not been found.

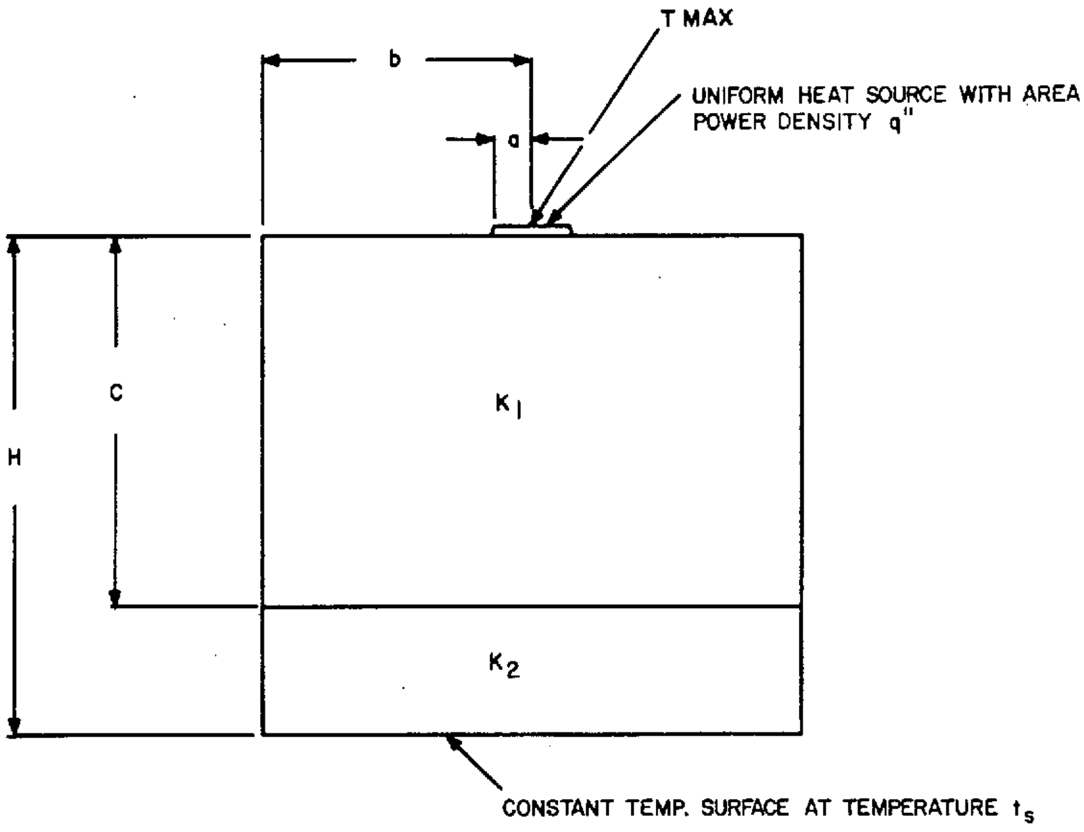
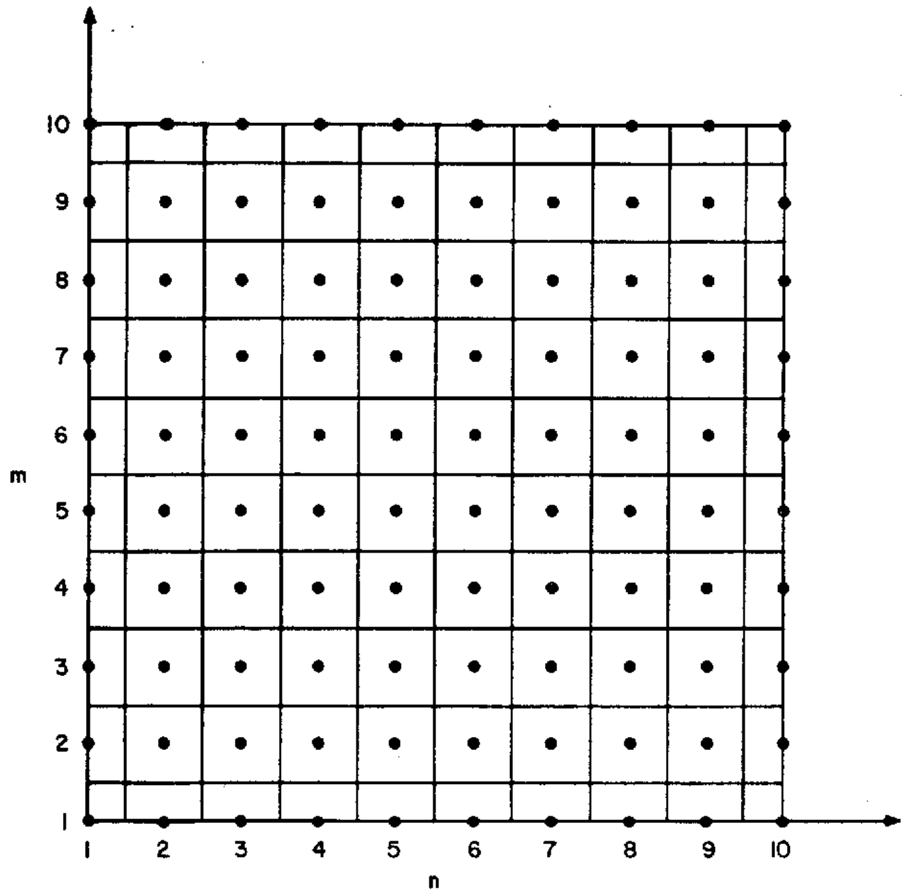


Figure 11. Mathematical Model for Spreading Resistance Nomographs



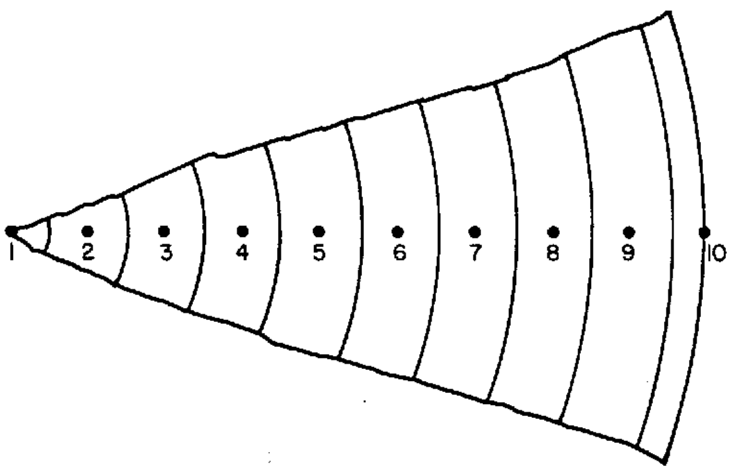


Figure 12. Nodal Pattern for Sample Problem

Ratios of geometries and conductivities used in the generation of sample data are summarized below. All data is generated and presented in nondimensional parameters.

Parameter	Number of Values	Parametric Values
a/b	6	0.111, 0.222, 0.166, 0.551, 0.388, 0.5
H/b	6	0.1, 0.2, 0.5, 1, 2, 5
C/H	5	0.1, 0.2, 0.3, 0.5,
K1/K2	5	0.01, 0.1, 0.2, 0.5, 1

The above three-dimensional problem can be viewed as two-dimensional since all heat flow within the cylinder is in the axial and radial directions. Further, there is symmetry about the axis of the cylinder.

Figure 12 shows the arrangement of finite elements used in the illustrative data generation program.

B. CALCULATION OF NODAL CONDUCTANCES

The convention used for the nodal conductances was that the vertical conductance $V_{(m,n)}$ associated with each node was that in the upward direction and the horizontal conductance $H_{(m,n)}$ was that connecting with the node on the right. This is illustrated in Figure 13.

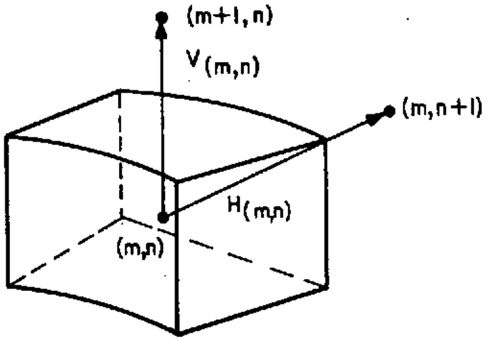


Figure 13. Convention for Nodal Conductances

The calculation of conductances is straightforward for all modes except for those nodes of horizontal conductance lying on the axis of the cylinder, i.e., n = 1. The horizontal conductance of this inner node was approximated by using the exact solution of Jacob (Ref. 9) for two-dimensional heat flow within a cylinder having uniformly distributed internal heat generation for the difference between the mean temperature of the cylinder and the outside surface with radial flow.

The conductances of all other nodes could be calculated in an exact manner using calculus. General expressions for the M, Nth nodal conductances in terms of M, N were developed and used in the sample problem to facilitate changing the program to allow the use of different numbers of finite elements.

C. DETAILED DESCRIPTION OF ILLUSTRATIVE PROGRAM

Figure 14 shows the computer program for the illustrative program. The major steps in the program are described below.

Line No.	Description
10-60	Dimensioning symbols will be defined as used in this program. The same symbol may be redefined several times.
70	P = π
80	C1 is that part of the horizontal thermal resistance of nodes 10, 1 and 1, 1 between the nodal point and the surface of these nodes. (See discussion above concerning Jacob's formula.)
90	Expressions for entire horizontal resistance between nodes 10, 1 and 10, 2 as well as 1, 1 and 1, 2.
110	H is the horizontal conductance.
160	V is the vertical conductance to node above.
200	Generalized expression for horizontal conductance for most modes, m, n.
210	Generalized expression for vertical conductance for most modes, m, n.
290, 320	Entering adiabatic boundary conditions at top and sides.
380-460	A9 is the radius of the heat generating region.
470	Calculation and print of a/b (see Figure 11).
480-250	Entering uniform heat input in the circular region described by A9. Use is made of the area dependence share by vertical conductance and heat input.
530-651	Setting of H/b (see Figure 11).
680-690	Adjustment of horizontal and vertical conductance for H/b.
720-801	Setting of C/H (see Figure 11).
810-971	Establishment and assignment of values for K1/K2 (see Figure 11).
990, 1000	Optional printout of $H_{(m, n)}$ and $V_{(m, n)}$
1010-1220	Generation of coefficient matrix.
1010	M is row number of physical nodal pattern.
1080	Sets subdiagonal and superdiagonal in the coefficient matrix (line arrays W immediately either side of the main diagonal) to zero (see Figure 15d).

(Continued on page 25)

```
1PRINT"    A/B          ";"H/B          ";"C/H          ";"K1/K2          ";"Y"
2 PRINT
10 DIM H(12,12),V(12,12),T(12,12),Q(10,10)
20 DIM X(10,10),Y(10,10),W(10,10),Z(10,10)
30 DIM A(10,10),B(10,10),D(10,10),E(10,10)
40 DIM F(10,10),G(10,10),I(10,10),J(10,10),K(10,10)
50 DIM L(10,10),M(10,10),N(10,10),Ø(10,10),P(10,10)
60 DIM R(10,10),S(10,10),U(10,10)
70 P=3.14159265
80 C1=.125/P
90 R1=C1+(LØG(2))/(2*P)
100 R2=R1/2
101 FØR N4=4 TØ 5
102 FØR N3=1 TØ5
103 FØR N2=1 TØ 6
104 FØR N1=1 TØ9
110 H(1,1)=H(10,1)=1/R1
120 FØR M=2 TØ 9
130 H(M,1)=2*H(1,1)
140 NEXT M
150FØR M=1 TØ 10
160 V(M,1)=P/2
170 NEXT M
180 FØR M=1 TØ 10
190 FØR N=2 TØ 10
200H(M,N)=1/(1/(4*P)*LØG(N/(N-1)))
210 V(M,N)=P*4*(N-1)
220 IF M>1.1 THEN 240
230H(1,N)=.5*H(1,N)
240 H(10,N)=.5*H(10,N)
250 V(M,10)=P*(4*N-5)/2
260 NEXT N
270 NEXT M
280 FØR M=1 TØ 10
290 H(M,10)=0
300 NEXT M
310 FØR N=1 TØ 10
320 V(1,N)=0
330 NEXT N
380 IF N1=1 THEN 2660
390 IF N1=5 THEN 2660
400 IF N1=7 THEN 2660
410 IF N1=9 THEN 2660
420 A9=1
430 IF N1<2 THEN 470
440 A9=2*N1-1
450 IF N1<10 THEN 470
460 A9=18
470 PRINT USING 471,A9/18,
471: ##.##
480 C(1,1)=V(2,1)*2.00000
490 IF N1=1 THEN 530
491 FØR N=2 TØ 10
492 C(N,1)=0
```

Figure 14. Computer Program for Illustrative Example
(Sheet 1 of 6)

```
493 NEXT N
500 FØR N=1 TØ N1
510 C(N,1)=V(2,N)*2.0000
520 NEXT N
530 H1=.1
540 IF N2<2 THEN 650
550 H1=.2
560 IF N2<3 THEN 650
570 H1=.5
580 IF N2<4 THEN 650
590 H1=1
600 IF N2<5 THEN 650
610 H1=2
615 IF N2< 6 THEN 650
620 H1=5
630 IF N2<7 THEN 650
640 H1=10
650 PRINT USING 651,H1,
651:########.##
660 FØR M=1 TØ 10
670 FØR N=1 TØ 10
680 V(M,N)=V(M,N)/H1
690 H(M,N)=H(M,N)*H1
700 NEXT N
710 NEXT M
720 C1=1
730 IF N3<2 THEN 800
740 C1=2
750 IF N3<3 THEN 800
760 C1=3
770 IF N3<4 THEN 800
780 C1=5
785 IF N3<5 THEN 800
790 C1=7
800 PRINT USING 801,C1/10,
801:########.##
810 K2=1
820 IF N4<2 THEN 910
830 K2=2
840 IF N4< 3 THEN 910
850 K2=5
860 IF N4<4 THEN 910
870 K2=10
880 IF N4<5 THEN 910
890 K2=100
910 FØR M=C1+1 TØ 10
920 FØR N=1 TØ 10
930 V(M,N)=V(M,N)*K2
940 H(M,N)=H(M,N)*K2
950 NEXT N
960 NEXT M
970 PRINT USING 972,1/K2,
972:##########.##
980 GØ TØ 1010
```

Figure 14. Computer Program for Illustrative Example
(Sheet 2 of 6)

```
990 MAT PRINT H;
1000 MAT PRINT V;
1010 FØR M=1 TØ 10
1020 MAT X=ZER
1030 MAT Y=ZER
1040 MAT W=ZER
1050 FØR N=1 TØ 10
1060 IF N=1 THEN 1090
1070 IF N=10 THEN 1090
1080 W(N,N-1)=W(N,N+1)=0
1090 IF N<2 THEN1110
1100 W(N,N-1)=-H(M,N-1)
1110 IF N> 9THEN 1130
1120 W(N,N+1)=-H(M,N)
1130 X(N,N)=-V(M+1,N)
1140 Y(N,N)=-V(M,N)
1150 C3=0
1160 IF N=1 THEN 1180
1170 C3=W(N,N-1)
1180 IF N=10 THEN 1200
1190 C3=C3+W(N,N+1)
1200 W(N,N)=C3+X(N,N)+Y(N,N)
1210 W(N,N)=-W(N,N)
1220 NEXT N
1230 MAT Z= INV(W)
1240 MAT W= Z*X
1250 MAT T=(-1)*W
1260 MAT W=T*(1)
1270 MAT X= Z*Y
1280 MAT T=ZER
1290 MAT T=(-1)*X
1300 MAT X=T*(1)
1310 IF M<>1 THEN 1350
1320MAT A=Z*C
1330 MAT C=ZER
1340 MAT B=W*(1)
1350 IF M<>2 THEN 1380
1360 MAT C=X*(1)
1370 MAT D=W*(1)
1380 IF M<>3 THEN 1410
1390 MAT E=X*(1)
1400 MAT F=W*(1)
1410 IF M<>4 THEN 1440
1420 MAT G=X*(1)
1430 MAT I=W*(1)
1440 IF M<>5 THEN 1470
1450 MAT J=X*(1)
1460 MAT K=W*(1)
1470 IF M<>6 THEN 1500
1480 MAT L=X*(1)
1490 MAT M=W*(1)
1500 IF M<>7 THEN 1530
1510 MAT N=X*(1)
1520 MAT Ø=W*(1)
```

Figure 14. Computer Program for Illustrative Example
(Sheet 3 of 6)

```
1530 IF M<>8THEN 1560
1540 MAT P=X*(1)
1550 MAT Q=W*(1)
1560 IF M<>9THEN 1590
1570 MAT R=X*(1)
1580 MAT S=W*(1)
1590 IF M<10 THEN 1630
1600 MAT T=ZER
1610 MAT T=X*(1)
1620 MAT U=W*(1)
1630 NEXT M
1640 FØR M=1 TØ 10
1650 FØR N=1 TØ 10
1660 W(M,N)=X(M,N)=Y(M,N)=Z(M,N)=0
1670 NEXT N
1680 NEXT M
1690 MAT X=C*B
1700 MAT Y=IDN
1710 MAT Z=Y-X
1720MAT X=INV (Z)
1730 MAT Z=X*D
1740 MAT D=Z*(1)
1750 MAT Z=C*A
1760 MAT C=X*Z
1770 MAT X= E*D
1780 MAT Z=Y-X
1790 MAT X=INV(Z)
1800 MAT Z=X*F
1810 MAT F=Z*(1)
1820 MAT Z=E*C
1830 MAT E=X*Z
1840 MAT X= G*F
1850 MAT Z=Y-X
1860 MAT X=INV(Z)
1870 MAT Z= X*I
1880 MAT I=Z*(1)
1890 MAT Z=G*E
1900 MAT G=X*Z
1910 MAT X=J*I
1920 MAT Z=Y-X
1930 MAT X=INV(Z)
1940 MAT Z=X*K
1950 MAT K=Z*(1)
1960 MAT Z=J*G
1970 MAT J=X*Z
1980 MAT X=L*K
1990 MAT Z=Y-X
2000 MAT X=INV(Z)
2010 MAT Z=X*M
2020 MAT M=Z*(1)
2030 MAT Z=L*J
2040 MAT L=X*Z
2050 MAT X=N*M
2060 MAT Z=Y-X
```

Figure 14. Computer Program for Illustrative Example
(Sheet 4 of 6)

```
2070 MAT X=INV(Z)
2080 MAT Z=X*Ø
2090 MAT Ø=Z*(1)
2100 MAT Z=N*L
2110 MAT N= X*Z
2120 MAT X=P*Ø
2130 MAT Z=Y-X
2140 MAT X=INV(Z)
2150 MAT Z=X*Q
2160 MAT Q=Z*(1)
2170 MAT Z=P*N
2180 MAT P=X*Z
2190 MAT X=R*Q
2200 MAT Z=Y-X
2210 MAT X=INV(Z)
2220 MAT Z=X*S
2230 MAT S=Z*(1)
2240 MAT Z= R*P
2250 MAT R=X*Z
2260 MAT X= T*S
2270 MAT Z=Y-S
2280 MAT X= INV (Z)
2290 MAT Z=X*U
2300 MAT U=Z*(1)
2310 MAT Z=T *R
2320 MAT T=ZER
2330 MAT X= ZER
2340 MAT X=S*T
2350 MAT Y=ZER
2360 MAT Y=X+R
2370 MAT R=Y*(1)
2380 MAT X=Q*Y
2390 MAT Y=X+P
2400 MAT P=Y*(1)
2410 MAT X=Ø*Y
2420MAT Y=X+N
2430 MAT N=Y*(1)
2440 MAT X=M*Y
2450 MAT Y=X+L
2460 MAT L=Y*(1)
2470 MAT X=K*Y
2480 MAT Y=X+J
2490 MAT J=Y*(1)
2500 MAT X=I*Y
2510 MAT Y=X+G
2520 MAT G=Y*(1)
2530 MAT X=F*Y
2540 MAT Y=X+E
2550 MAT E=Y*(1)
2560MAT X=D *Y
```

Figure 14. Computer Program for Illustrative Example
(Sheet 5 of 6)

```
2570 MAT Y=X+C
2580 MAT C=Y*(1)
2590 MAT X=B*Y
2600 MAT Y=X+A
2610 MAT A=Y*(1)
2630 R7=A(1,1)/A9
2640 PRINT USING 2650,R7
2650 :####.##
2660 NEXT N1
2665 PRINT
2670 NEXT N2
2675 PRINT
2680 NEXT N3
2685 PRINT
2690 NEXT N4
2695 PRINT
2700 END
```

Figure 14. Computer Program for Illustrative Example
(Sheet 6 of 6)

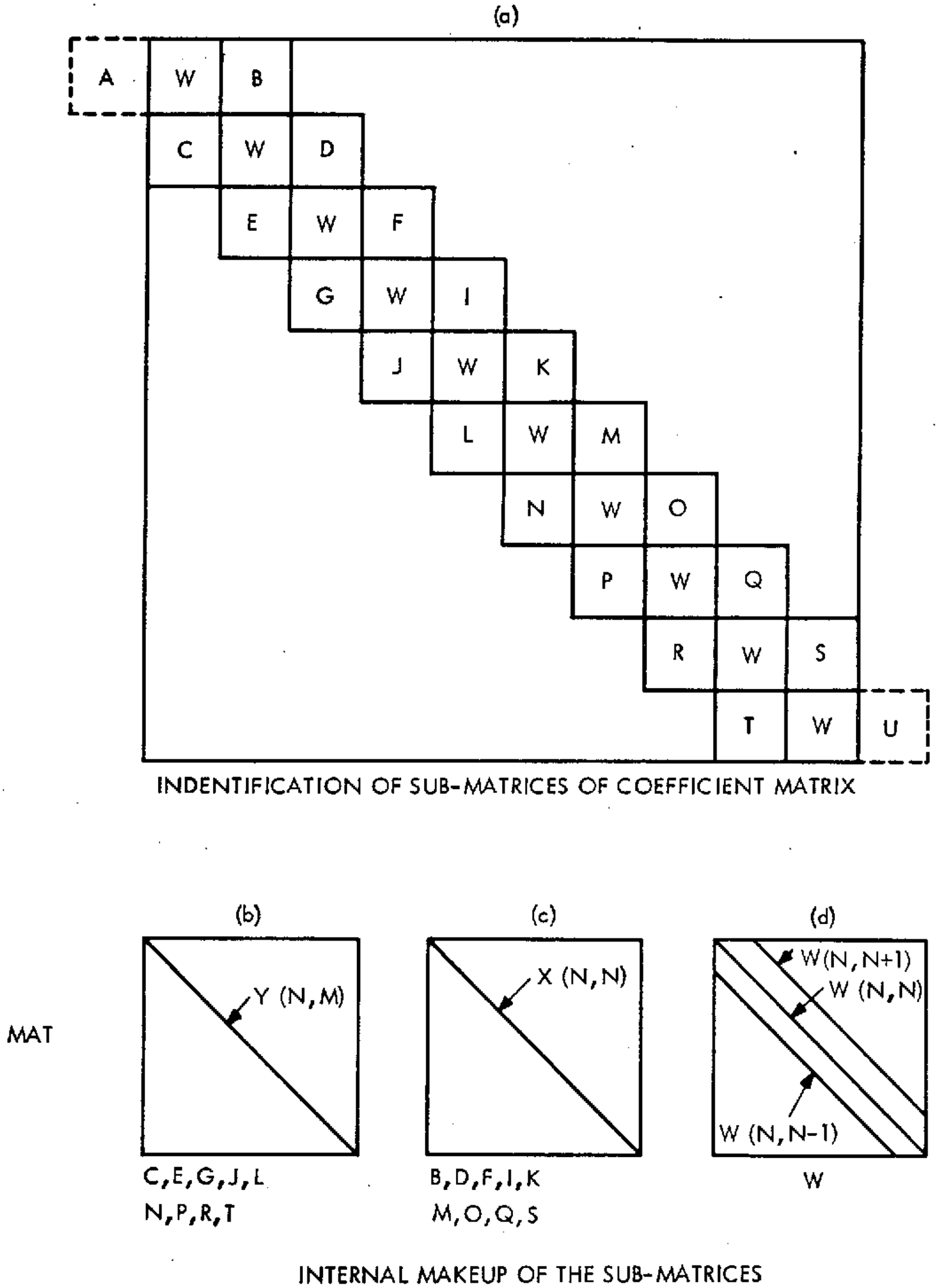


Figure 15. Identification and Internal Makeup of Submatrices of Coefficient Matrix

Line No.	Description
1100	Assigns values to first subdiagonal of the coefficient matrix W (see Figure 15d) describing the mth row of the physical nodal pattern.
1120	Assigns values to first subdiagonal of the coefficient matrix W as in line 1100.
1130	Assigns values to the X submatrix (see Figure 15c).
1140	Assigns values to the Y submatrix (see Figure 15b).
1200, 1210	Enters elements down the main diagonal of the W submatrices.
1230	First submatrix manipulation statement.
1240	Normalizes the submatrix to the right of the W matrix diagonal (see Figure 15a).
1270	Normalizes the submatrix to the left of the W matrix (see Figure 15a).
1280	Empties T matrix.
1320-1630	Assigns each of the submatrices, subdiagonal and superdiagonal, mapped by Figure 15a after normalization. Note that matrices X and W are functions of M, the row of the physical nodal pattern.
1640-1680	Clears matrices W, X, Y, Z, so they may be redefined below by entering only nonzero elements.
1690-2250	These steps calculate the recurrence coefficients A_i' and B_i' according to equation (10) of Section II. As these are calculated, the superdiagonal and subdiagonal matrices of Figure 15a are sequentially redefined to be these recurrence coefficients.
2320	This statement enters the boundary condition that the temperatures along the bottom row of the physical matrix are zero.
2330-2610	Using the recurrence relationship developed in 1690-2250, equation (10) is used to calculate the temperatures, one row (of the physical nodal model) at a time. These (column) matrices of temperatures are calculated in the following order (see Figure 15a) and with the following nomenclature: T, R, P, N, L - J, G, E, C, A.
2630	R7 is the temperature of node 1, 1 divided by the radius of the heat source A9, defined in statements 420-460.

D. SAMPLE SOLUTIONS

Figure 16 presents the sample solutions of the illustrative problem. t_{MAX} is the average temperature nodal point 1, 1, see Figure 12. It is obvious that, for these solutions, increasing the number of nodes in the physical model would have the result of increasing the temperature in this hottest element. This was done and the results are discussed below.

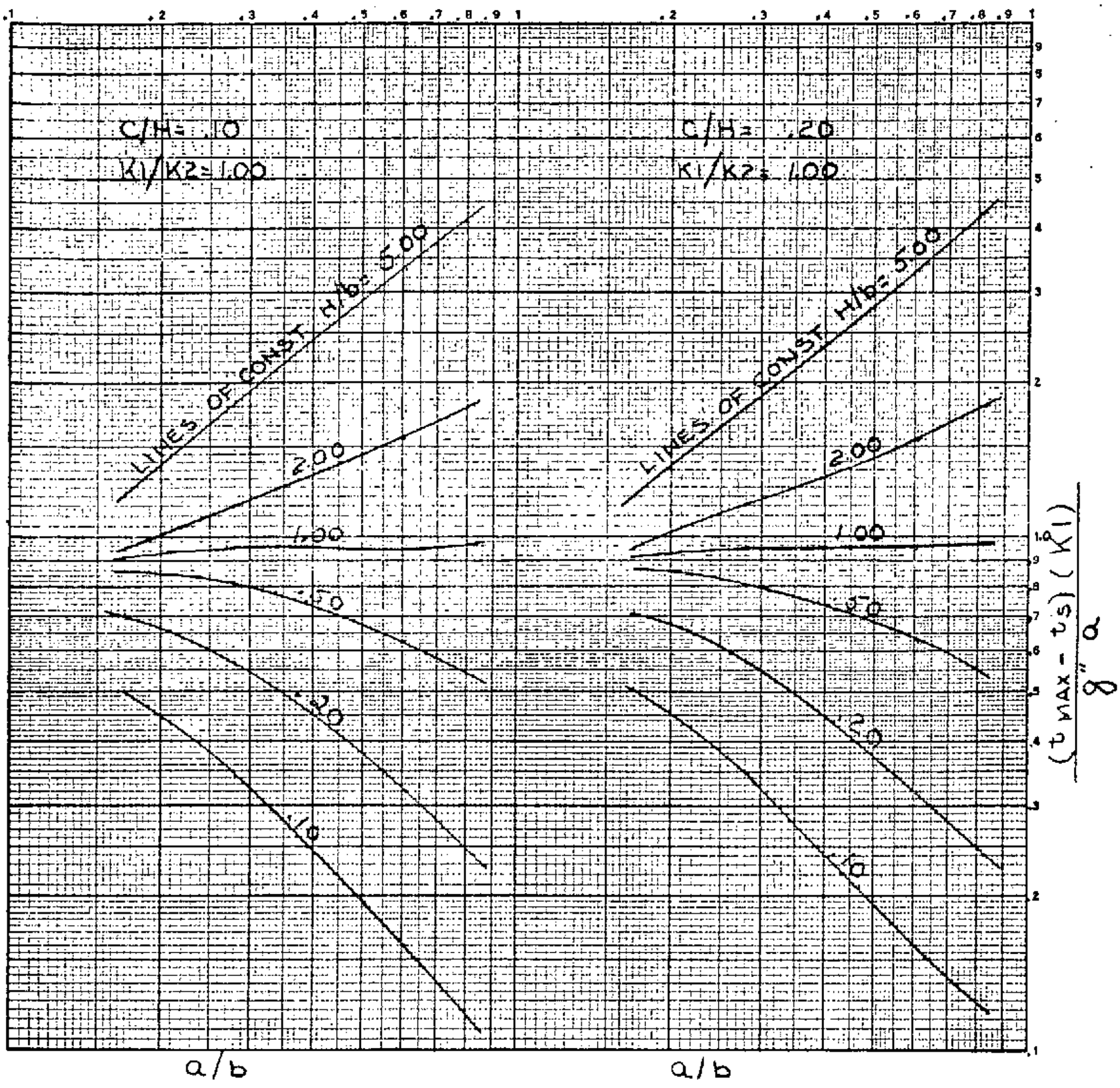


Figure 16. Sample Solution of Illustrative Problem
(Sheet 1 of 10)

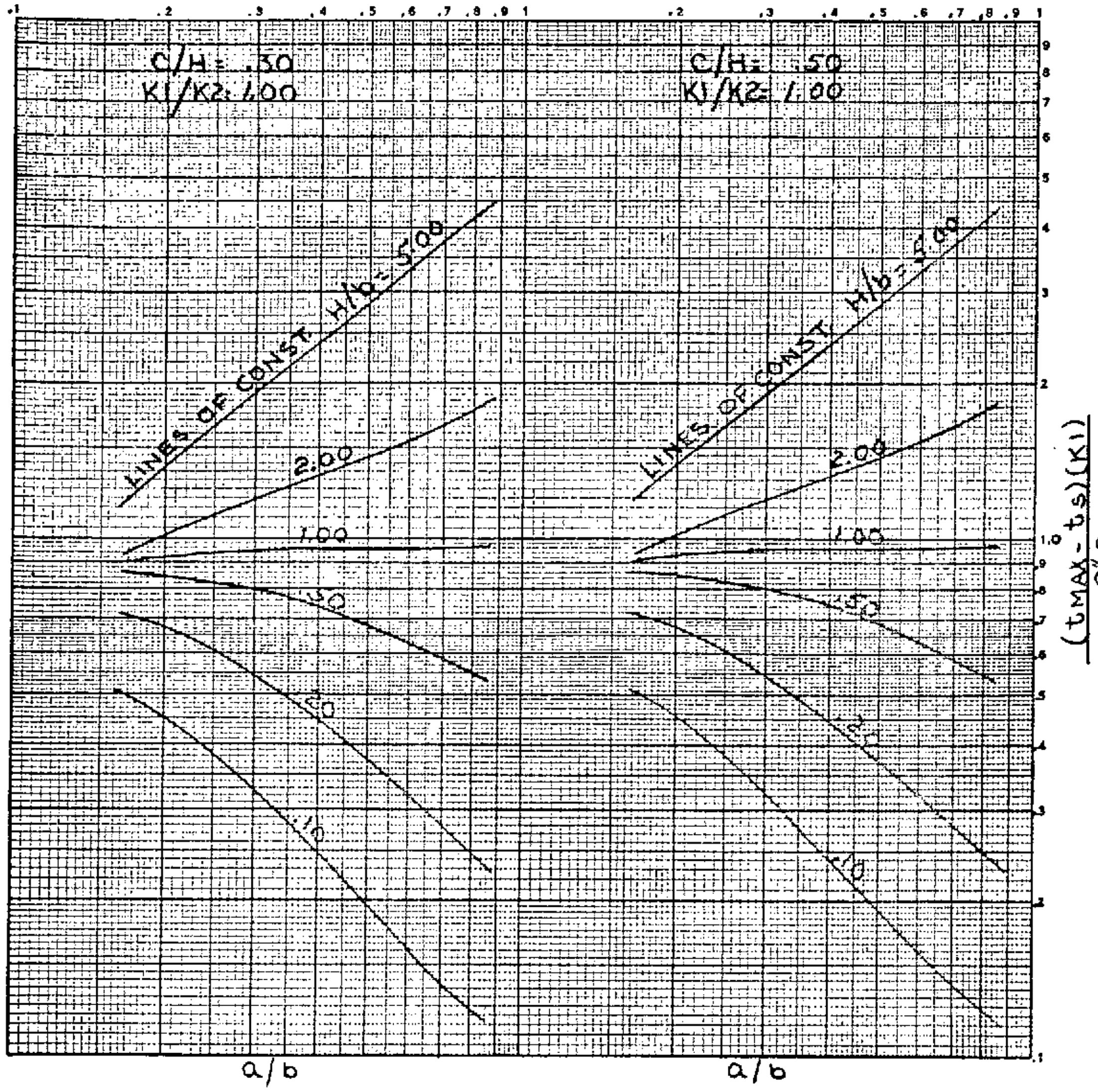


Figure 16. Sample Solution of Illustrative Problem
(Sheet 2 of 10)

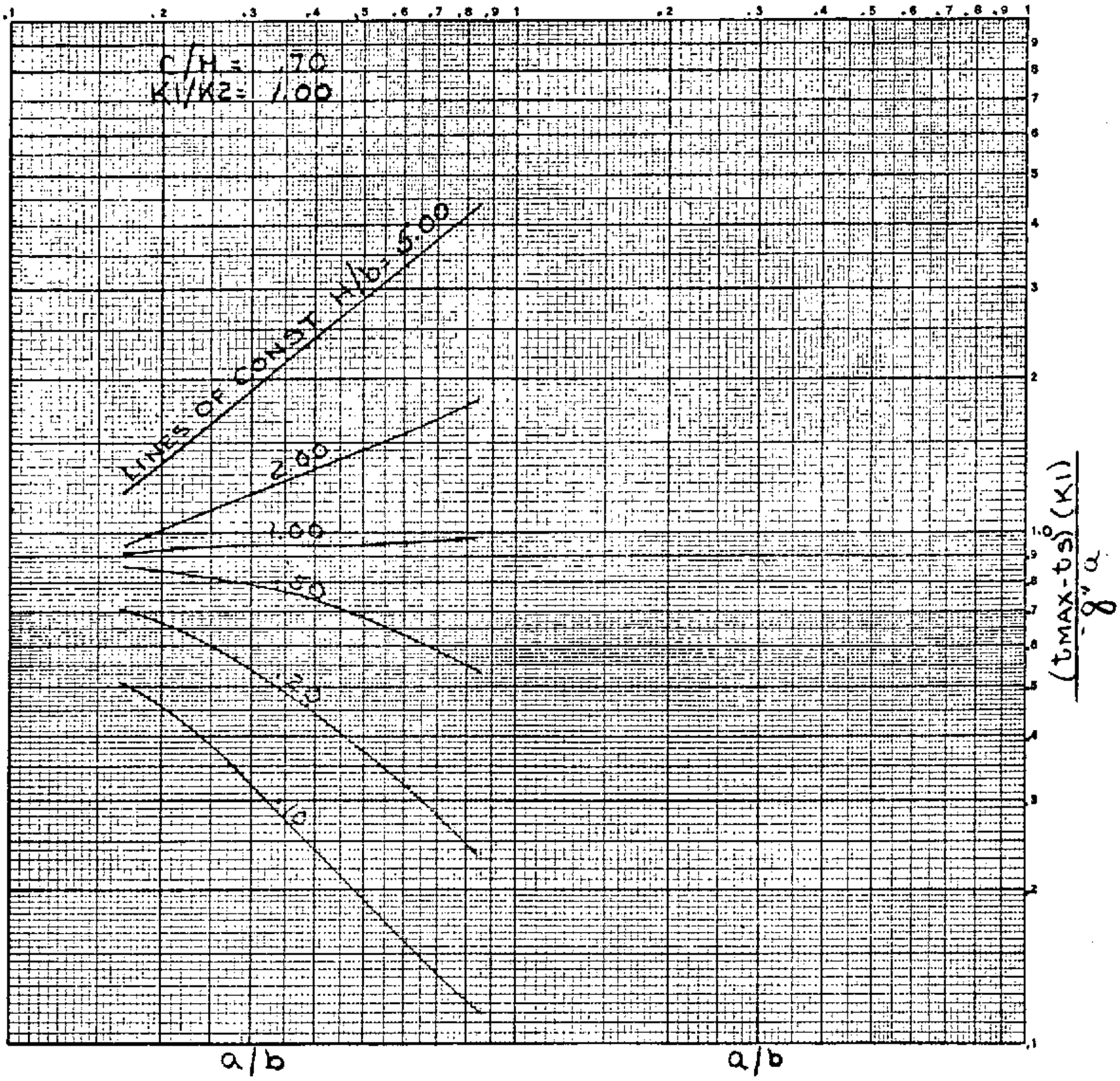


Figure 16. Sample Solution of Illustrative Problem
(Sheet 3 of 10)

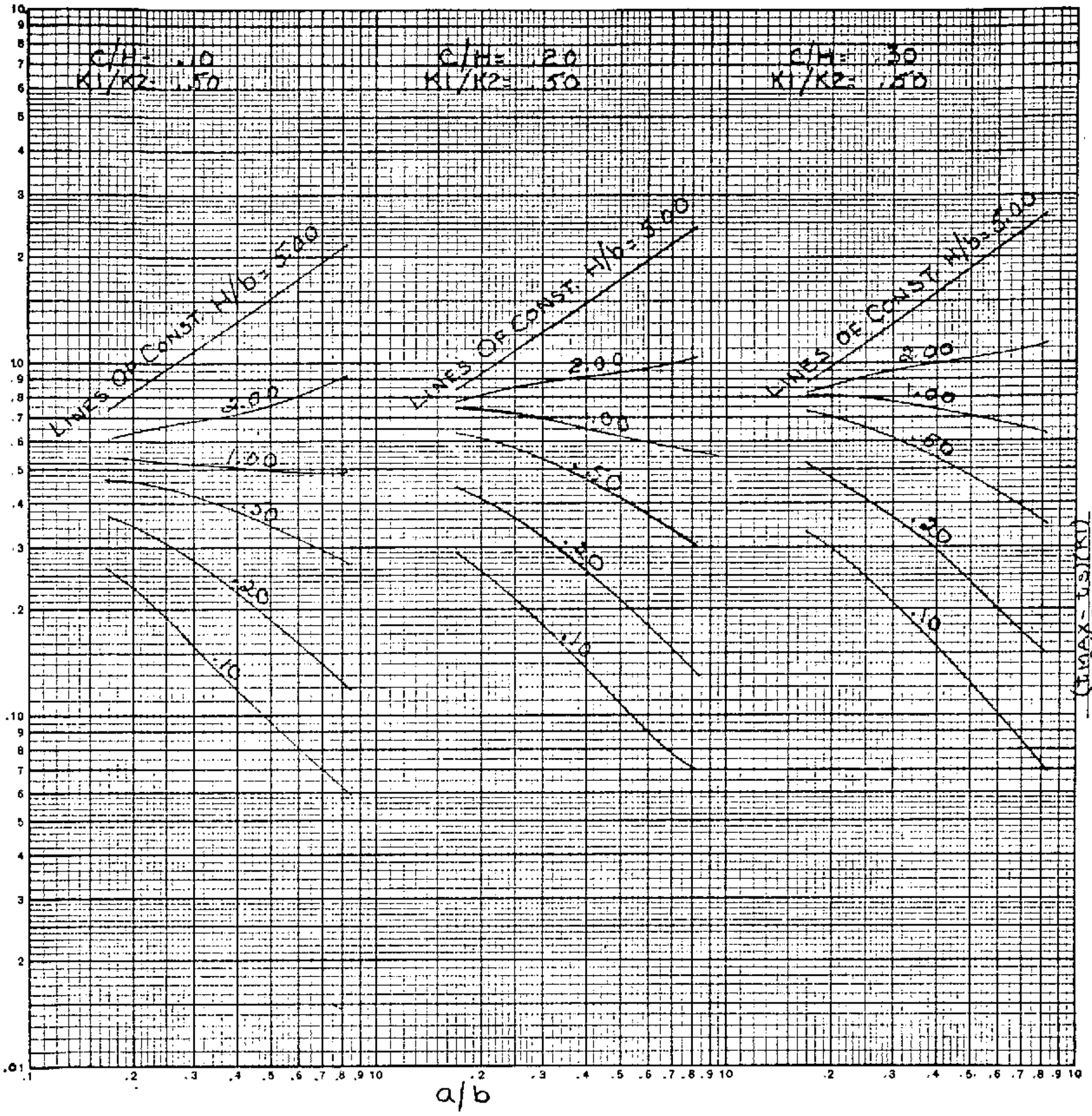


Figure 16. Sample Solution of Illustrative Problem
(Sheet 4 of 10)

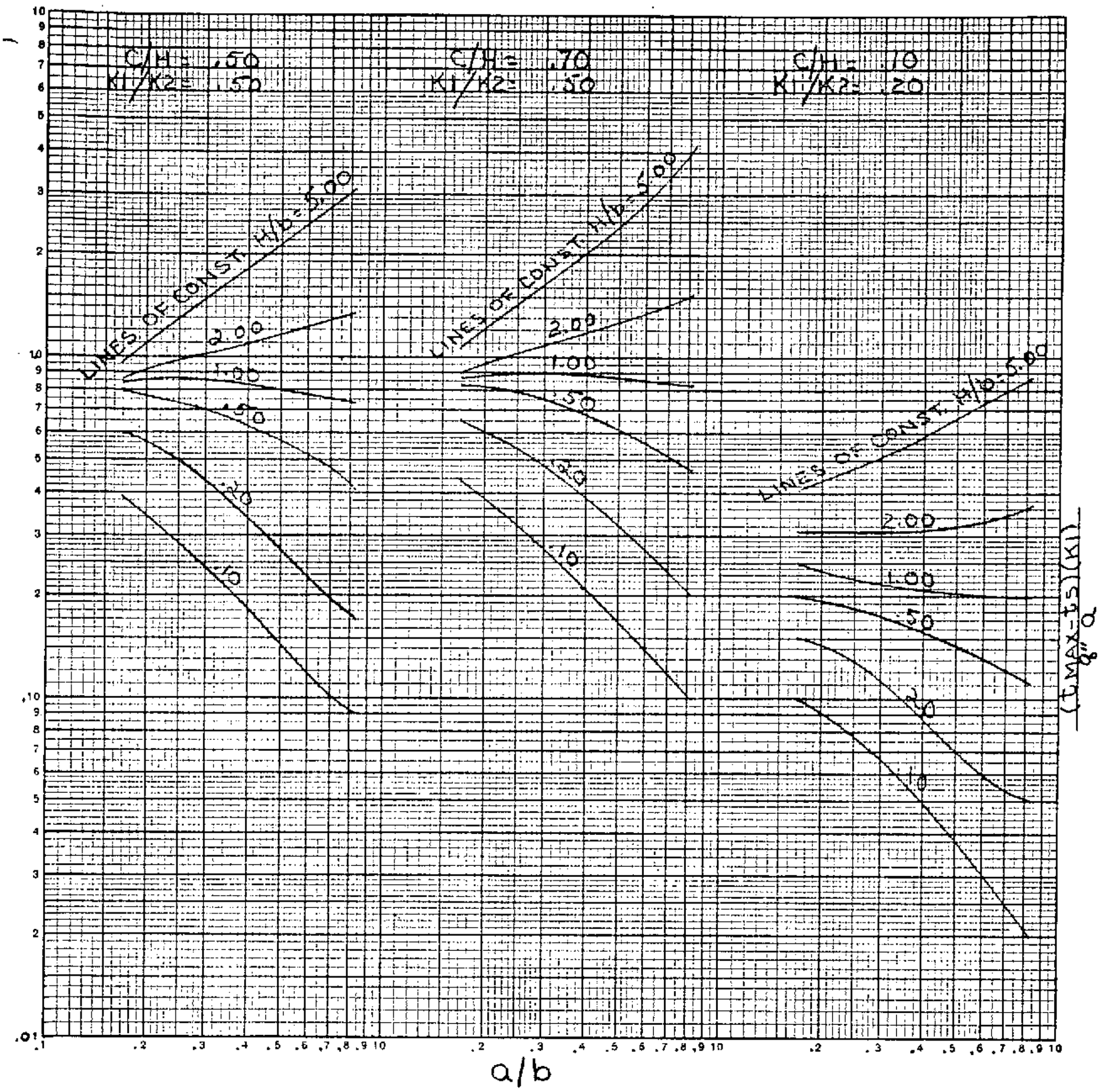


Figure 16. Sample Solution of Illustrative Problem
(Sheet 5 of 10)

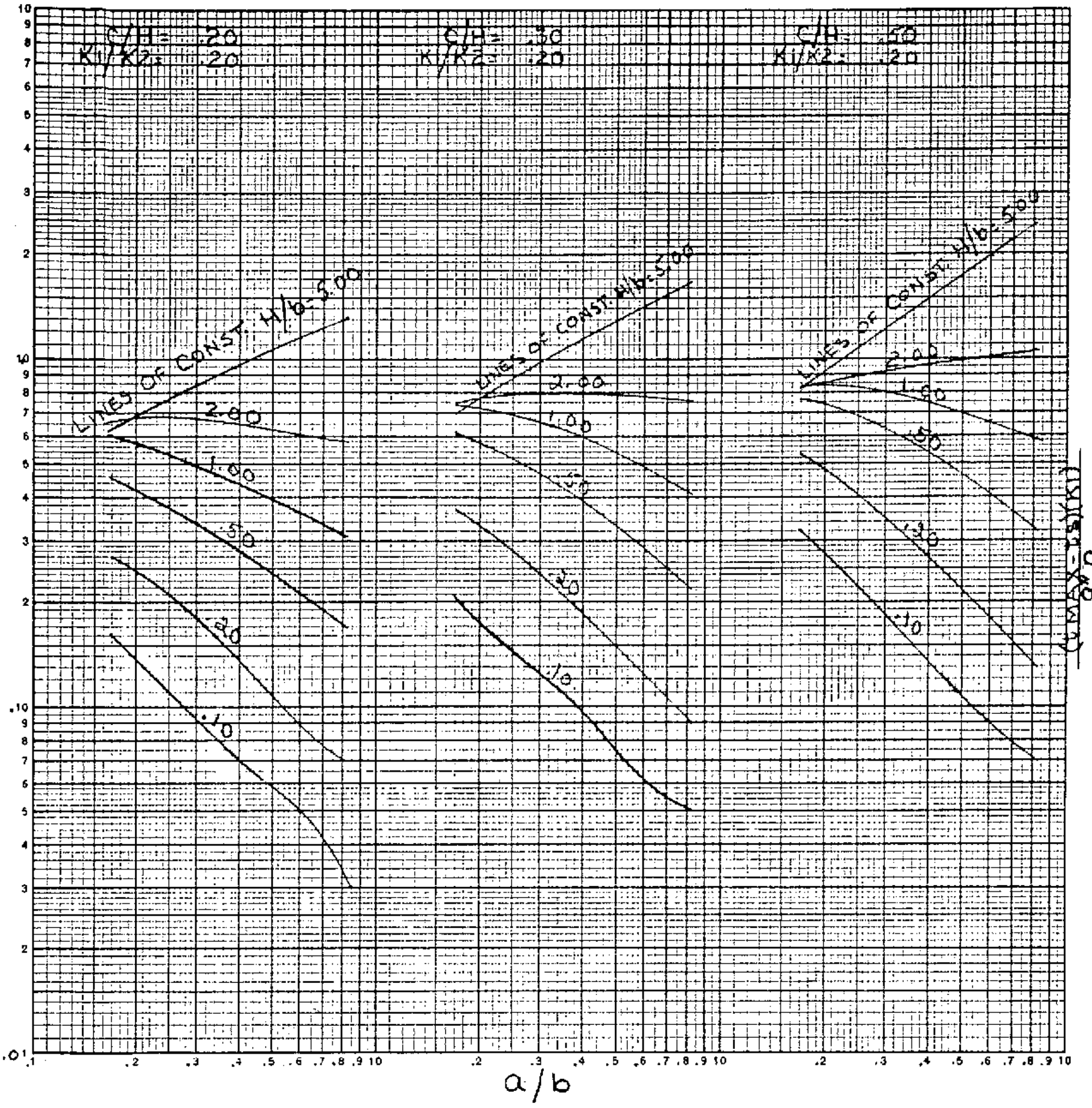


Figure 16. Sample Solution of Illustrative Problem
(Sheet 6 of 10)

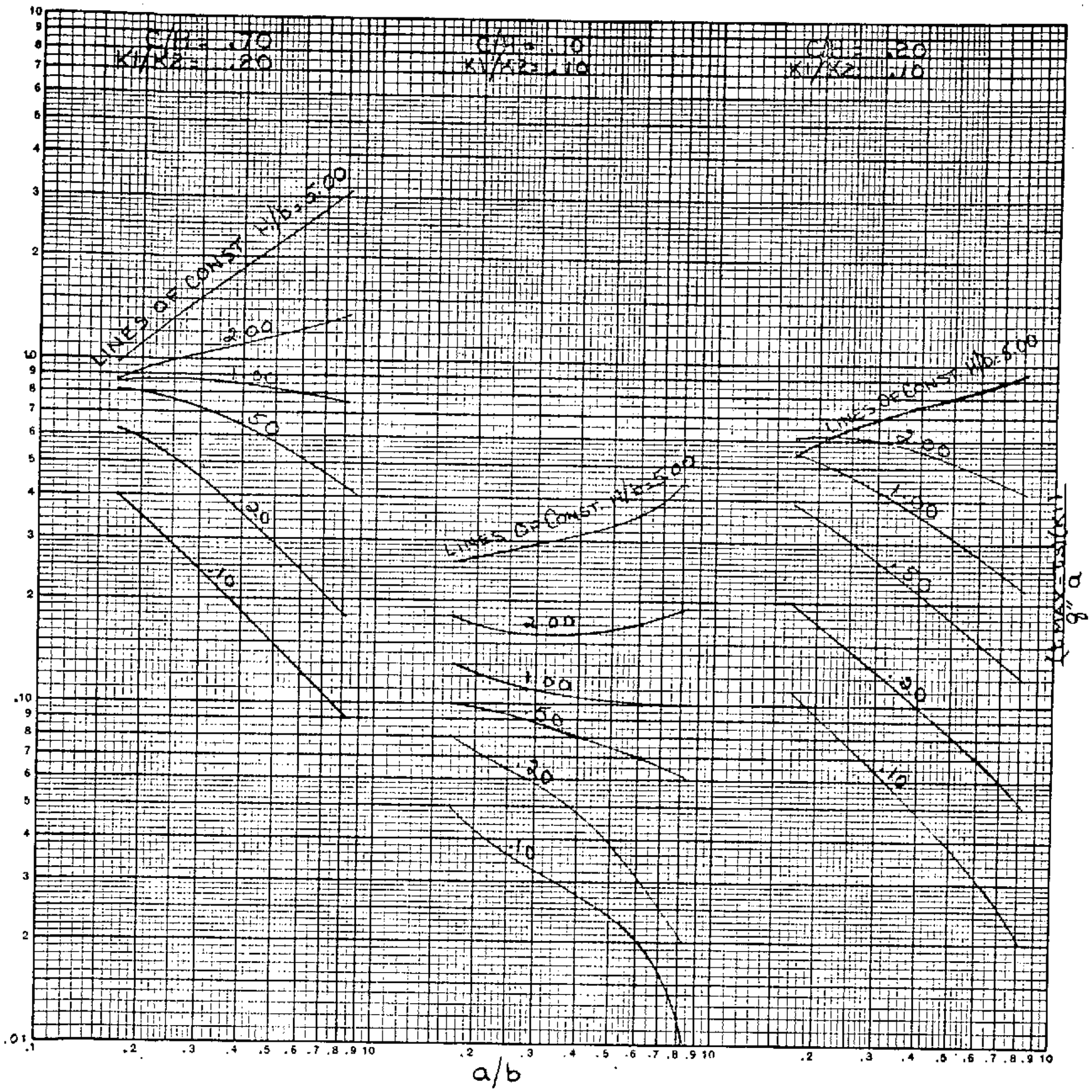


Figure 16. Sample Solution of Illustrative Problem
(Sheet 7 of 10)

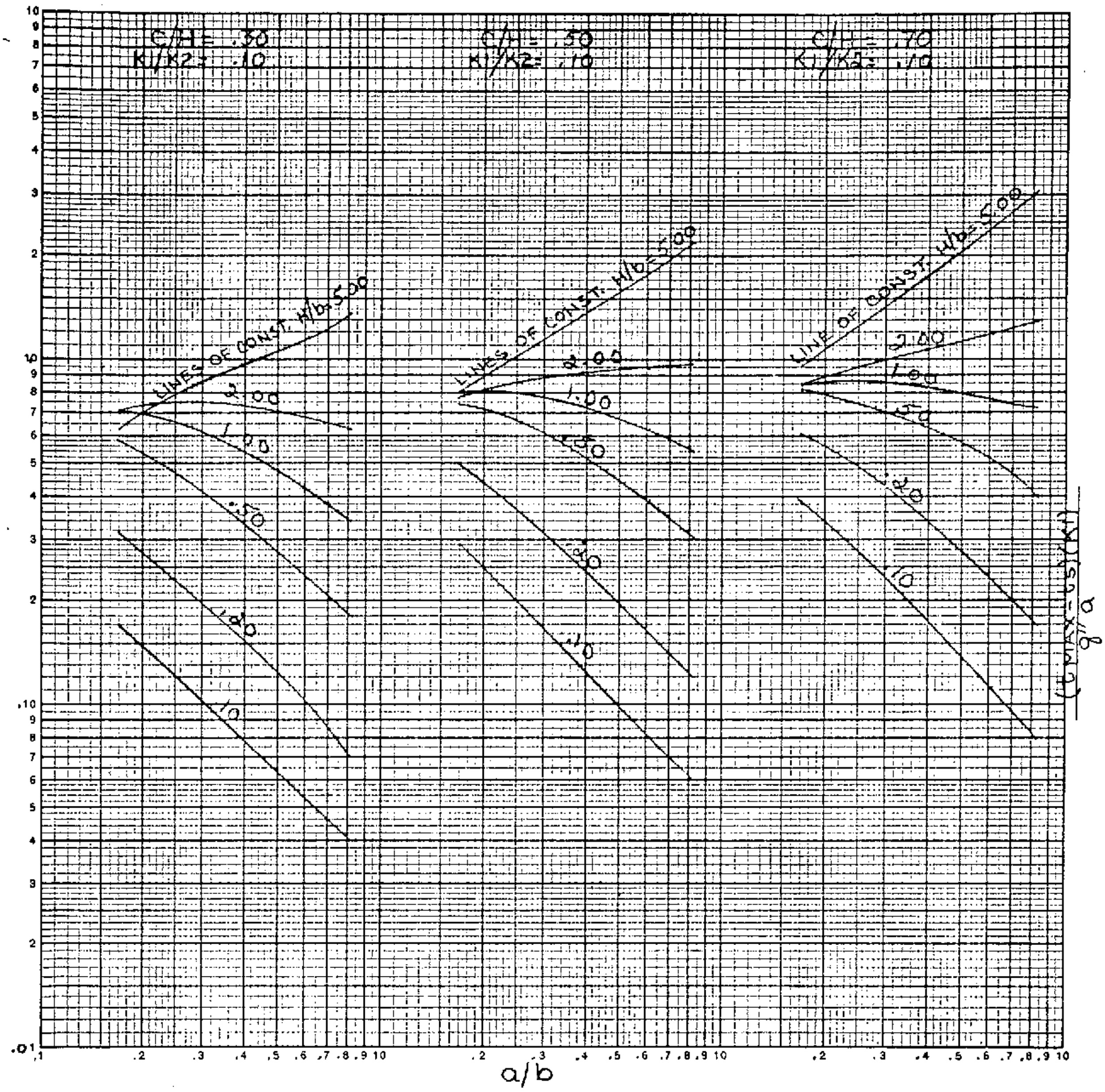


Figure 16. Sample Solution of Illustrative Problem
(Sheet 8 of 10)

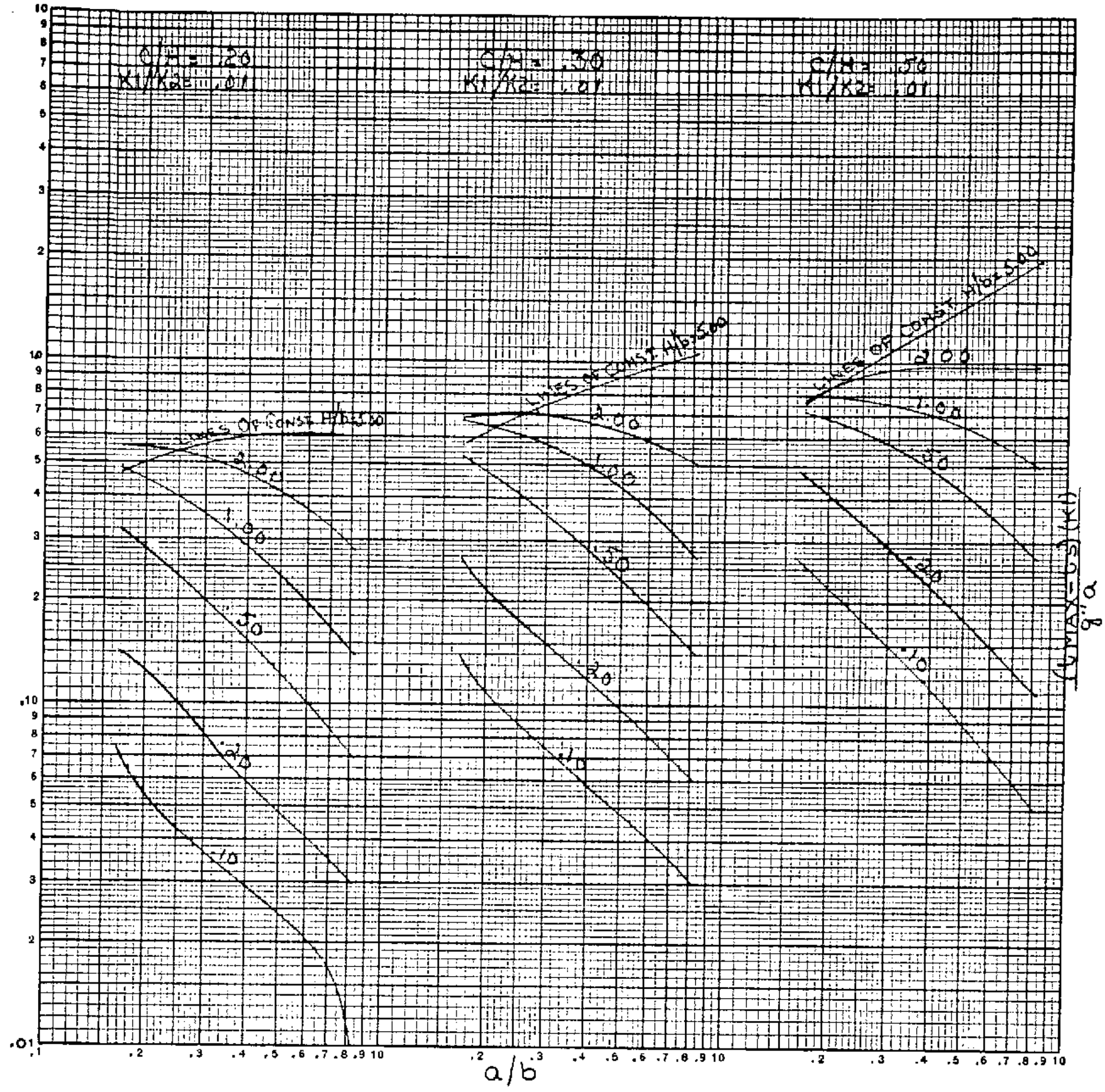


Figure 16. Sample Solution of Illustrative Problem
(Sheet 9 of 10)

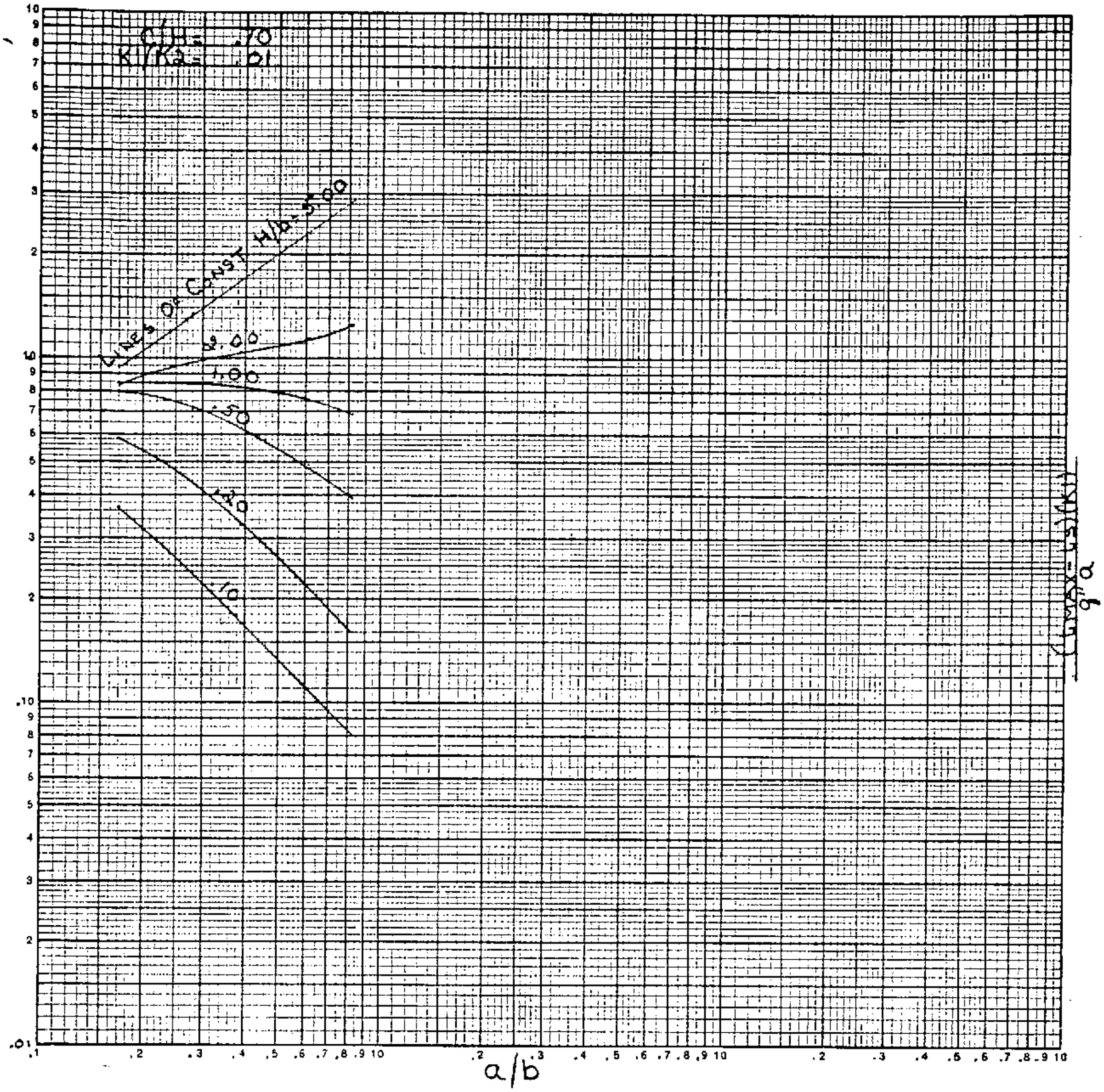


Figure 16. Sample Solution of Illustrative Problem
(Sheet 10 of 10)

Effect of Increasing the Number of Finite Elements on the Solution of the Illustrative Problem

An exact closed form solution exists for the cylindrical spreading resistance problem in a medium of uniform conductivity. Kennedy (Ref. 10) shows that, as $a/b \rightarrow 0$,

$$\frac{(t_{MAX} - t_s)\,K1}{q''\,a} \rightarrow 1$$

Figure 17 shows this trend for the finite difference model. With 1600 nodes, $(t_{MAX}-t_S)\,K1/q''$ was calculated to be 0.9788. With 2500 nodes, the nondimensional resistance dropped to 0.9766; this reduction is attributed to the inexact treatment of the horizontal resistance of the nodes in the inner column using the equation of Jacob (Ref. 9) as described earlier.

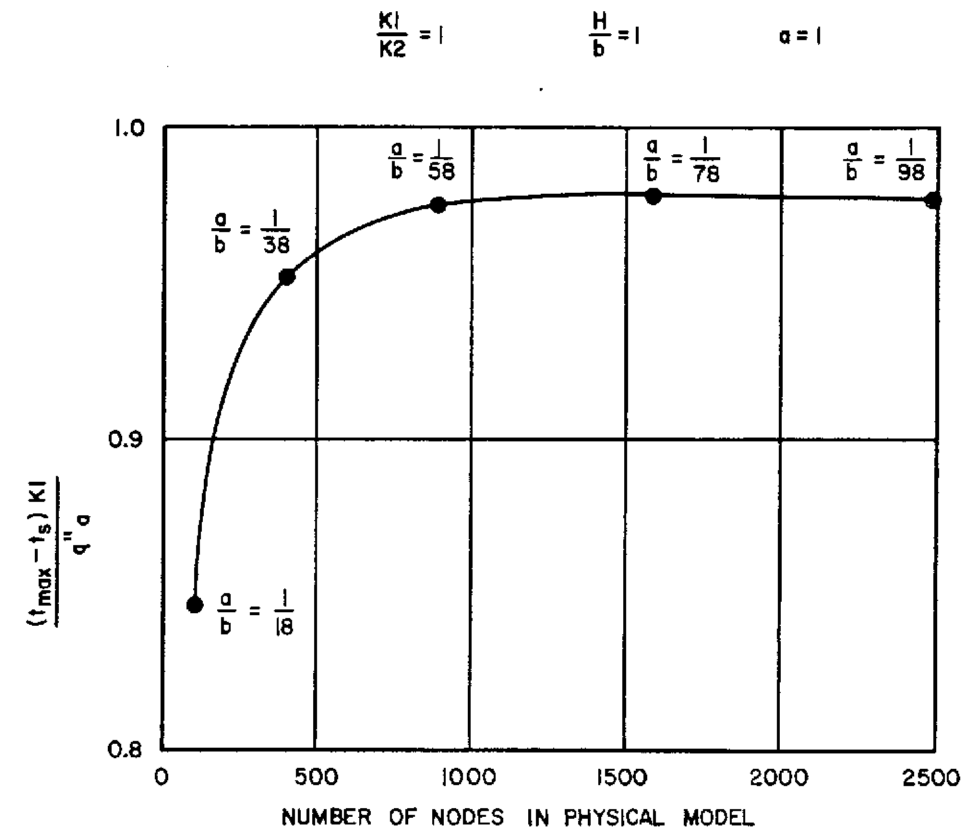


Figure 17. Maximum Nodal Temperature for Minimum a/b as a Function of Number of Nodes

It would appear that a nodal model of 900 nodes would be a near optimum number for generating solutions to this particular problem using the finite element approach.

SECTION V

THEORY OF SUPERPOSITION OF SOLUTIONS OF SPREADING THERMAL RESISTANCE PROBLEMS

A. GENERAL

The general steady state heat-conduction equation in Cartesian coordinates, known as the Poisson equation, is given by

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{\partial^2 t}{\partial z^2} + \frac{q'''}{K} = 0 \qquad (14)$$

where

t = $t(x, y, z)$ = temperature (° F)

K = thermal conductivity (taken to be independent of temperature and position) (Btu/hr-ft-° F)

q''' = $q'''(x, y, z)$ = internal volumetric heat source (Btu/hr-ft^3)

x, y, z = Cartesian coordinates

The generalized boundary conditions vary. For example, specified temperature:

$$t(x, y, z)\Big|_{x_b, y_b, z_b} = f(x_b, y_b, y_b) \qquad (15a)$$

where:

f = specified function

x_b, y_b, z_b = values of x, y, z on the boundary (b)

or, convection to a fluid:

$$\frac{\partial t}{\partial n}\Big|_{x_b, y_b, z_b} = \frac{h}{\underline{\underline{k}}}\left[t(x, y, z)\Big|_{\text{on the boundary}} - t_f \right] \qquad (15b)$$

where

n = outward directed vector normal to the boundary

h = Newtonian convective film coefficient (Btu/hr-ft^2-° F)

t_f = bulk temperature of convecting fluid (° F)

It is convenient to rewrite equations (14), (15a), and (15b) using a temperature difference for the dependent variable that contains a reference temperature. This reference temperature is typically taken to be a specified boundary temperature, as in equation (15a), or the fluid temperature as in equation (15b).

Hence, we define this temperature difference by

$$u \triangleq t - t_{\text{reference}} \tag{16}$$

Equations (14), (15a), and (15b) can then be rewritten

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + G = 0 \tag{17}$$

$$u(x, y, z)\Big|_{x_b, y_b, z_b} = g(x_b, y_b, z_b) \tag{18a}$$

$$\frac{\partial u}{\partial n}\Big|_{x_b, y_b, z_b} = \frac{h}{K}\, u(x_b, y_b, z_b) \tag{18b}$$

where

$$G = q'''/K$$

These equations are linear as can be seen by observing that they contain no products of the dependent variable (u) or its derivatives. Since they are linear, any linearly independent combination of solutions will satisfy these equations due to the distributive property of linear operators, i.e.,

$$L(x_1 + x_2 + \ldots) = L(x_1) + L(x_2) + \ldots$$

where

L is a generalized linear operator

This property may be applied to equation (17) as follows: Take u_1 and u_2 to be independent solutions to equation (17). Next, define

$$u_3 = a_1 u_1 + a_2 u_2$$

where $a_1 u_1 + a_2 u_2 = 0$ if, and only if, $a_1 = a_2 = 0$, i.e., u_1 and u_2 are linearly independent.

Now substitute $a_1 u_1$ into equation (17), then $a_2 u_2$ into equation (17), and add the two expressions [using (a) the shorthand operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

and (b) G_1 corresponding to the $a_1 u_1$ solution and G_2 corresponding to the $a_2 u_2$ solution]:

$$\nabla^2 a_1 u_1 + \nabla^2 a_2 u_2 + G_1 + G_2 = 0$$

Since ∇^2 is a linear operator, and defining $G = G_1 + G_2$, we can write:

$$\nabla^2(a_1 u_1 + a_2 u_2) + G = 0$$

but

$$a_1 u_1 + a_2 u_2 = u_3$$

therefore,

$$\nabla^2 u_3 + G = 0$$

Thus, u_3 is also a solution to equation (17). Applying the same procedure to, say, equation (18b)

$$\frac{\partial u_1}{\partial n} = \frac{h}{K} u_1$$

$$\frac{\partial u_2}{\partial n} = \frac{h}{K} u_2$$

Adding

$$\frac{\partial u_1}{\partial n} + \frac{\partial u_2}{\partial n} = \frac{h}{K} (u_1 + u_2)$$

Since $\partial/\partial n$ is a linear operator, this can be written

$$\frac{\partial}{\partial n} (u_1 + u_2) = \frac{h}{K} (u_1 + u_2)$$

But $u_1 + u_2 = u_3$, then

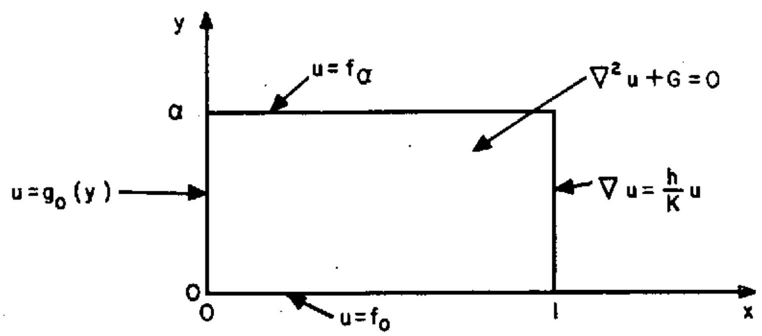
$$\frac{\partial u_3}{\partial n} = \frac{h}{K} u_3$$

Thus, u_3 also satisfies the boundary condition.

B. ADDITIVE SOLUTIONS

A useful ramification of the superposition principle lies in the fact that the solution to a complicated system may be formed by linear combinations of known solutions.

Consider a rectangular plate with internal heat generation. The governing differential equation and boundary conditions are illustrated in the following sketch:



Explicitly, we have the system

$$
\begin{aligned}
\nabla^2 u(x,y) + G(x,\ y) &= 0 \\
u(x,0) &= f_o(x) \\
u(x,\alpha) &= f_\alpha(x) \\
u(0,y) &= g_o(y) \\
\nabla u(1,y) &= (h/K)\, u(1,\ y)
\end{aligned}
\tag{19}
$$

The solution may now be written

$$
u(x,y) = u_1(x,y) + u_2(x,y) + u_3(x,y) + u_4(x,y) \tag{20}
$$

The number of ancillary problems is taken equal to the number of nonhomogeneities in the system. Since the governing equation and the first three boundary conditions are not homogeneous, the number of ancillary problems is four.

Substituting equation (20) into equation (19), we obtain the complete system:

$$
\begin{aligned}
\nabla^2 u_1 + \nabla^2 u_2 + \nabla^2 u_3 + \nabla^2 u_4 + G &= 0 \\
u_1 + u_2 + u_3 + u_4 &= f_o \\
u_1 + u_2 + u_3 + u_4 &= f_\alpha \\
u_1 + u_2 + u_3 + u_4 &= g_o \\
\nabla u_1 + \nabla u_2 + \nabla u_3 + \nabla u_4 &= \left(\frac{h}{K}\right)\left[u_1 + u_2 + u_3 + u_4\right] \qquad \text{at } x = 1,\ y = y
\end{aligned}
\tag{21}
$$

The set of ancillary problems corresponding to this system can now be written as:

Problem 1	Problem 2	Problem 3	Problem 4
$\nabla^2 u_1 + G = 0$	$\nabla^2 u_2 = 0$	$\nabla^2 u_3 = 0$	$\nabla^2 u_4 = 0$
$u_1 = 0$	$u_1 = 0$	$u_1 = 0$	$u_1(x,0) = f_o(x)$
$u_2 = 0$	$u_2(x,\alpha) = f_\alpha(x)$	$u_2 = 0$	$u_2 = 0$
$u_3 = 0$	$u_3 = 0$	$u_3(0,y) = g_o(y)$	$u_3 = 0$
$\nabla u_1(1,y) = \frac{h}{K}\, u_1(1,y)$	$\nabla u_2(1,y) = \frac{h}{K}\, u_2(1,y)$	$\nabla u_3(1,y) = \frac{h}{K}\, u_3(1,y)$	$\nabla u_4(1,y) = \frac{h}{K}\, u_4(1,y)$

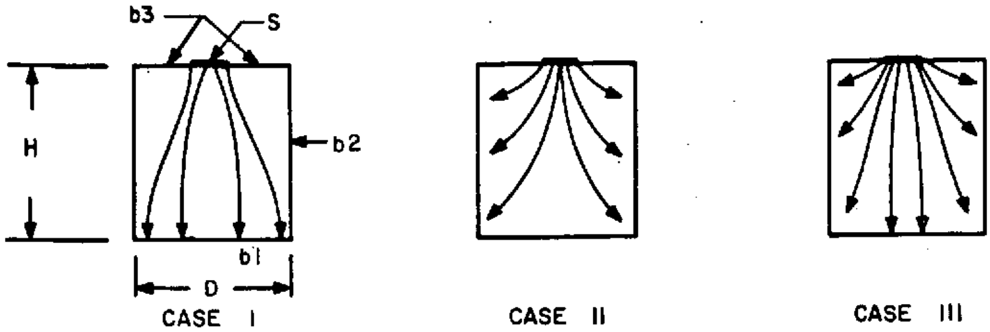
Note that each of the ancillary problems now contains only one nonhomogeneity, a much simpler form. Note also that their sum is equal to equations (21).

Hopefully, we can solve each of the ancillary problems or find the solutions in the literature. Once we have these, we merely add them to obtain the total solution to equations (19).

It was mentioned that once a system has been degenerated to a set of ancillary problems, the final solution is the sum of the individual solutions. This holds, provided the ancillary

problems are properly defined. Care must be exercised in specifying boundary conditions so that the sum of the individual solutions equals the total solution.

Consider, for example, the following three problems (taken from reference 10):



Heat, which is generated uniformly over a circular disk S, spreads by conduction through a cylinder of height H and diameter D to a constant temperature heat sink.

At first glance, the analyst might be inclined to assume that case III is the sum of cases I and II; he would be wrong. To understand why, we must correctly define the boundary conditions on each ancillary problem.

The governing differential equation in each case will be the same

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{22}$$

The boundary conditions will be written in four parts corresponding to the regions b1, b2, b3 and S.

<u>Case I:</u>

$$\begin{aligned} u_I(b1) &= 0 \\ \frac{\partial u_I(b2)}{\partial n} &= 0 \text{ (adiabatic surface)} \\ \frac{\partial u_I(b3)}{\partial n} &= 0 \\ \frac{\partial u_I(S)}{\partial n} &= G_I \text{ (constant flux)} \end{aligned} \tag{23a}$$

where n is an outward directed unit vector normal to the surface.

Case II:

$$\frac{\partial u_{II}(b1)}{n} = 0$$

$$u_{II}(b2) = 0 \tag{23b}$$

$$\frac{\partial u_{II}(b3)}{\partial n} = 0$$

$$\frac{\partial u_{II}(S)}{\partial n} = G_{II}$$

Case III:

$$u_{III}(b1) = 0$$

$$u_{III}(b2) = 0 \tag{23c}$$

$$\frac{\partial u_{III}(b3)}{\partial n} = 0$$

$$\frac{\partial u_{III}(S)}{\partial n} = G_{III}$$

Now, add the governing differential equations for cases I and II

$$\frac{\partial^2 u_I}{\partial x^2} + \frac{\partial^2 u_I}{\partial y^2} + \frac{\partial^2 u_{II}}{\partial x^2} + \frac{\partial^2 u_{II}}{\partial y^2} = 0$$

Regrouping

$$\frac{\partial^2}{\partial x^2}(u_I + u_{II}) + \frac{\partial^2}{\partial y^2}(u_I + u_{II}) = 0$$

By our initial assumption $u_{III} = u_I + u_{II}$. Thus,

$$\frac{\partial^2 u_{III}}{\partial x^2} + \frac{\partial^2 u_{III}}{\partial y^2} = 0$$

The differential equation is satisfied.

Next, add boundary conditions, beginning in region b1:

$$u_I(b1) + \frac{\partial u_{II}(b1)}{\partial n} = 0$$

Note that we have mixed conditions which are inconsistent. This situation also exists in regions b2 and S (except that in region S we could define $G_I + G_{II} \triangleq G_{III}$ to make the boundary conditions additive. Thus, case III is not the sum of cases I and II; we have, in fact, three nonanalogous systems.

C. FURTHER EXAMPLES

An excellent set of examples of the application of superposition principles in the calculation of thermal spreading resistances may be found in Reference 1 in which the auther utilizes Green's function in the calculation of thermal spreading resistances. For a discussion of Green's function see Reference 11.

SECTION VI

SUMMARY OF OTHER TECHNIQUES FOR CALCULATING AND ESTIMATING THERMAL SPREADING RESISTANCE

The two handiest "rules of thumb" relationships that can be used to calculate or estimate thermal spreading resistances were developed by Holm (Ref. 12) and Raillard (Ref. 13). Holm derives the equation for the thermal resistance of a circular isothermal source on the face of a semi-infinite slab as:

$$R = \frac{1}{4\,a\,k} \tag{24}$$

where

a = radius of circular source

k = thermal conductivity of medium

Figure 3 (Section II) shows the temperature profiles described by Holm's equation. Figure 3 shows that 80 percent of the total resistance in a semi-infinite slab occurs within three radii of the source. It can be seen that, when the size of the source is small compared to the thickness of a slab of finite extent, Holm's equation can be used to make a conservative (high) estimate of thermal resistance. Such geometries occur often in microelectronic components.

Raillard presents similar exact solutions for circular and rectangular sources having uniform generation located on the faces of semi-infinite slabs. The equation for the thermal spreading resistance of the circular source bears a close resemblance to Holm's equation:

$$R = \frac{1}{\pi\,a\,k} \tag{25}$$

In Appendix B of his report, Raillard presents an exact closed-form solution for the rectangular source of uniform generation on a semi-infinite slab. He also derives the solution for uniform generation in a strip of infinite length and finite width on a semi-infinite slab.

Two references treat the case of the rectangular source of finite width and infinite length on a slab of finite depth. Wilcox (Ref. 14) treats the uniform heat generation source, while Gale (Ref. 15) presents thermal spreading resistances for the uniform temperature source. The results of these two studies can also be found in Reference 16.

Muller (Ref. 17) and Kennedy (Ref. 10) present exact solutions to the problem of a circular source at one end of a right cylinder with conduction to the side of the cylinder, the other end of the cylinder, or to both places. Kennedy's source is uniform while the intensity of Muller's circular heat source varies exponentially within the source region.

Hein (Ref. 1) examines stady-state heat transfer in a rectangular substrate or slab having multiple heat sources. He has integrated circuits in mind. He considers convective heat transfer and lead conduction for various heat-sinking conditions and his solutions are mathematically exact.

Finally, it might be well to point out a principle which is somewhat analogous to Saint-Venant's Principle in the theory of elasticity. Simply stated, T, the temperature in any thermal spreading resistance problem, varies with 1/L where L is the distance from the source, when L is large compared with the characteristic demension of the source. That is,

$$T \propto \frac{1}{L} \quad \text{when } L \gg a \text{ where a is characteristic source dimension}$$

This is indicated by the form of the solution of Fourier's equation for spherical flow, i.e.,

$$q = k\frac{A\,dt}{dt} = \frac{k_m\,(t_1 - t_2)}{\int_{L_1}^{L_2} \frac{r^2\,dr}{4\pi r^2}} = \frac{4\pi k_m\,(t_1 - t_2)}{\left(\frac{1}{L_2} - \frac{1}{L_1}\right)}$$

SECTION VII

PLANNED APPLICATION OF COMPUTATIONAL TECHNIQUE

General Electric's Aerospace Electronic Systems Department has developed an extremely compact and thermally efficient packaging configuration for computer circuitry. Integrated circuits are bonded to multilayered printed wiring boards which are in turn bonded to compact forced-air-cooled heat exchangers. This configuration is illustrated in Figures 18 and 19.

A heat transfer analysis program that calculates the temperature of each flatpack using Gauss-Seidel iteration is currently employed. This technique has proven costly: 200^{+} iterations are required. Computer costs of \$25 to \$50 per analysis have been experienced.

The exact computational technique described in this report will be implemented in the near future for this type of analysis. Costs are expected to be an order of magnitude lower than those with the iterative technique. (See Figure 9 of Section III.)

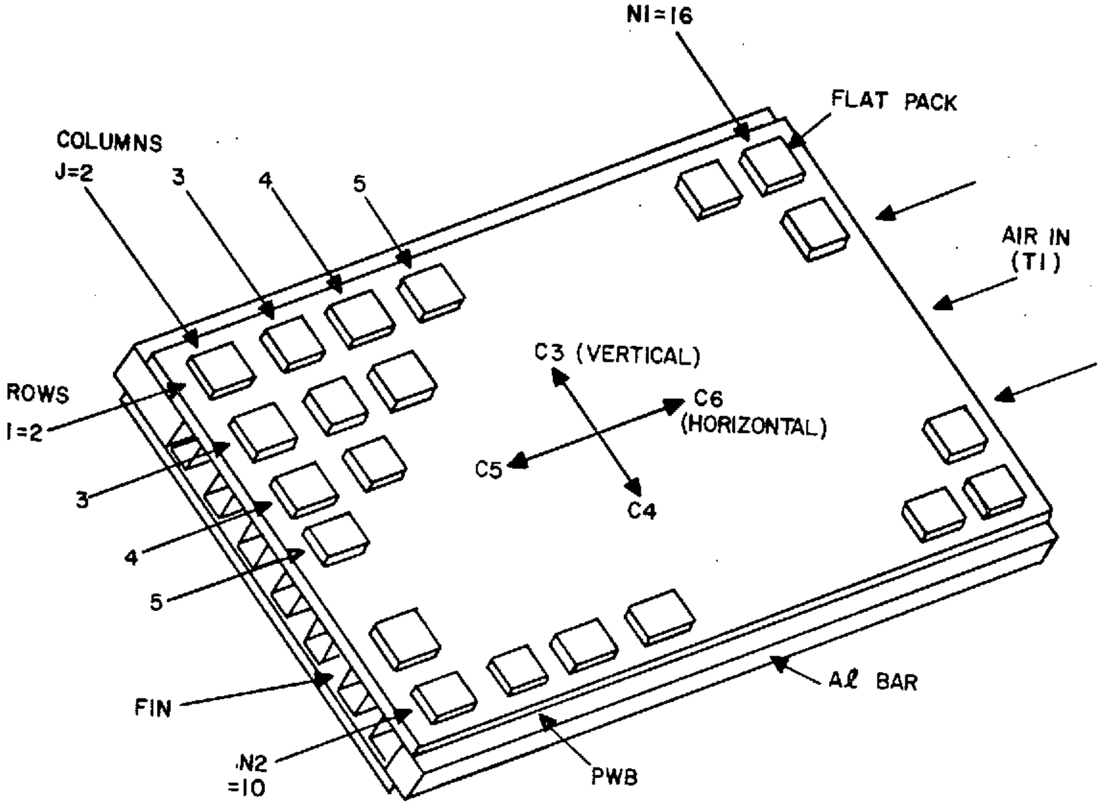


Figure 18. Memory Board Module

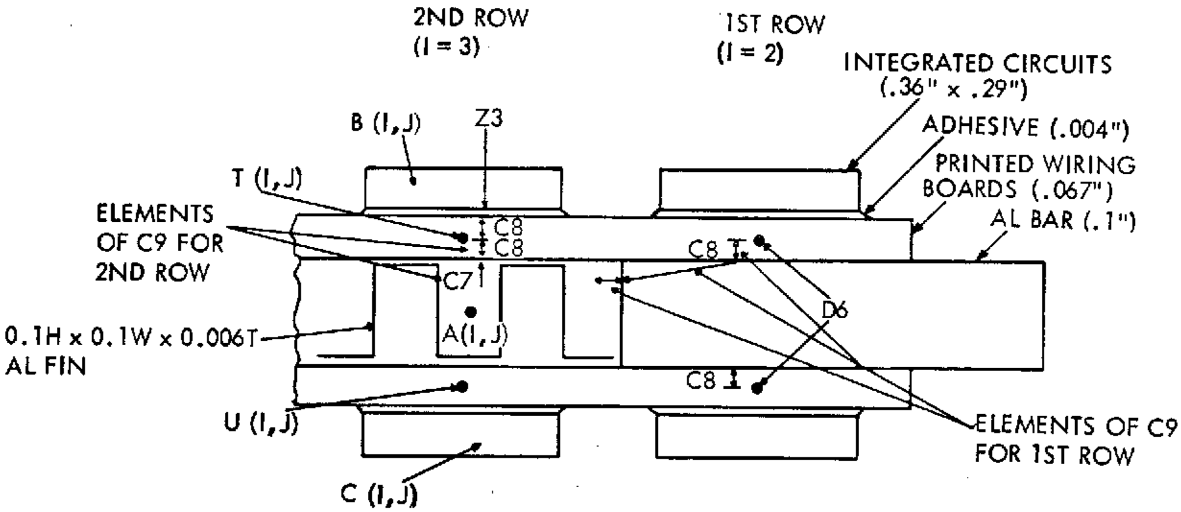


Figure 19. Location of Conductances and Temperatures

SECTION VIII

RECOMMENDATIONS FOR FUTURE WORK

1. The program should be rewritten for three-dimensional fields. Special attention to such techniques as using the method developed in this report for the main coefficient matrix on the submatrices themselves should be examined. The fact that very large matrices can be efficiently handled by this technique should be capitalized upon.

2. The program as it now stands should be used in a thermal spreading resistance parametric study similar to but larger in scope than the illustrative problem of this study. Nine hundred to sixteen hundred finite elements should be used in such data generation.

REFERENCES

1. Hein, V. L. "Convection and Conduction Cooling of Substrates Containing Multiple Heat Sources," The Bell System Technical Journal, XLVI, Number 8, October 1967.

2. Clausing, A.M., and Chao, B.T., "Thermal Contact Resistance in a Vacuum Environment," University of Illinois, PhD Disertation, 1963.

3. Tenech, H., and Rohsenow, W.M., "Reduction of Thermal Conductance of Metallic Surfaces in Contact, "ASME Paper No. 62-HT-32, 1962.

4. Gale, E.H., "Effects of Thin Surface Films on The Thermal Contact Resistance of Microscopically Sized Contacts," Syracuse University, PhD Desertation, 1970.

5. Mikic, B., and Carnasciali, G., "The Effect of Thermal Conductivity of Plating Material on Thermal Contact Resistance," ASME Journal of Heat Transfer, August 1970.

6. Karlqvist, O. Tellus, 4 (1952), p. 374.

7. Westlake, Joan R., A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, New York: John Wiley and Sons, 1968, p. 99.

8. Cornock, A.V., "The Numerical Solution of Poisson's and the Biharmonic Equations by Matrices," Proc. Cambridge Philosophical Society, Book 50, 1954, p. 524.

9. Jacob, M., Archiv fur Electrotechnik, 8, 1919, pp. 117-126.

10. Kennedy, D.P., "Spreading Resistance in Cylindrical Semiconductor Devices," 31, No. 8, August 1960, pp. 1490-1497.

11. Kaplan, W., Advanced Calculus, New York: Addison-Wesley, 1959, p. 665.

12. Holm, Ragnar, Electric Contacts - Theory and Application, 4th ed., New York: Springer-Verlag, 1967.

13. Raillard, H. "Heat Flow and Thermal Properties of Gallium-Arsenide Junction Lasers"; Syracuse University, Desertation, 1965 (may be obtained from University Microfilm Service, Ann Arbor, Michigan, Thesis No. 65-7978).

14. Wilcox, W.R., "Heat Transfer in Power Transistors," IEEE Trans. on Electron Devices, Sept. 1963, pp. 308-313.

15. Gale, E.H., "Thermal Conductance of Thin Film Resistors," Electronic Packaging and Production, 7, No. 5, May 1967, pp. 148-157.

16. References 15 and 16 may be found in General Electric Heat Transfer Design Data Book, Section G502-4, p. 15, January 1968 (this book is both in print and in open literature).

17. Muller, Alex, "On the Input Limit of an X-Ray Tube with Circular Focus," Proc. Royal Society of London, 117, Series A, 1927-28, pp. 30-42.

APPENDIX

GENERAL FORTRAN Y VERSION OF COMPUTER PROGRAM

```
$       IDENT    727-9C4,KELLY NED             ,DEL-B        65078300044700
$       OPTION   FORTRAN
$       USE      MEMORY/1000/
$       ENTRY    MAIN
$       FORTY
$       INCODE   IBMF
CMAIN            MAIN
      SUBROUTINE MAIN
      PARAMETER MAXCOR=1000
      COMMON /MEMORY/CORE(MAXCOR)
      COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
      COMMON /TIMES/ITIME(2),IDATE(2),DELTIM
      COMMON /DEBUG/IDEBUG
      PARAMETER MAXMAT=69
      PARAMETER MAXOFF=6
      DIMENSION IOFF(MAXOFF)
      DIMENSION KARD(14)
      PARAMETER MAXTIT=12
      DIMENSION ITITLE(MAXTIT)
      PARAMETER MAXTIM=2
      DATA KDEBUG/5HDEBUG/
      DATA IDEBUG/1/
      DATA INFILE/05/
      DATA IOFILE/06/
      DATA IFILE1/07/
      DATA JFILE1/6H000007/
      DATA IFILE2/08/
      DATA ISIZE/4HSIZE/
      DATA ICORE/MAXCOR/
      DATA KTITLE/6HTITLE /
      CALL FXOPT(67,1,1,0)
      CALL FXOPT(68,0,0,0)
      CALL FXOPT(69,0,0,0)
      CALL FXOPT(70,0,0,0)
      CALL FXOPT(71,0,0,0)
      READ(INFILE,77)KARD
      IF(KARD(1).NE.KTITLE) GO TO 94
      CALL SUPERT(MAXTIT,KARD(2))
   94 CONTINUE
      CALL NEWLIN
      WRITE(IOFILE,78)KARD
      CALL TIMDAT(ITIME,IDATE)
      CALL ELTIME(DELTIM)
```

```
  558 CONTINUE
      READ(INFILE,77)KARD
   77 FORMAT(13A6,A2)
      CALL NEWLIN
      WRITE(IOFILE,78)KARD
   78 FORMAT(2H *,13A6,A2,1H*)
      IF(KARD(1).NE.KDEBUG) GO TO 557
      IDEBUG=0
      GO TO 558
  557 CONTINUE
      DECODE(KARD,1)KEY,N
    1 FORMAT(A6,4X,I5)
      IF(KEY.EQ.ISIZE) GO TO 2
      CALL NEWLIN
      WRITE(IOFILE,55)
   55 FORMAT(40H THE ABOVE CARD SHOULD BE A 'SIZE' CARD.)
      STOP
    2 IF(N.LT.MAXMAT) GO TO 222
      CALL NEWLIN
      WRITE(IOFILE,223)MAXMAT
  223 FORMAT(18H SIZE GREATER THAN,I10)
      STOP
  222 IF(N.GT.0) GO TO 224
      CALL NEWLIN
      WRITE(IOFILE,225)
  225 FORMAT(17H SIZE LESS THAN 1)
      STOP
  224 NSQ=N*N
      CALL SETNUM(2*N+3)
C     RECORDS ARE IN SYSTEM STANDARD RANDOM FORMAT
C     WHICH MEANS THAT IF A RECORD IS GREATER THAN 318 WORDS
C                     THEN THE RECORD WILL BEGIN IN A NEW BLOCK
C                     AND END A BLOCK EVERY TIME
C
C     I TRIED USING PURE DATA RANDOM FILES(11-13-72), BUT
C     FOR SOME REASON THEY DID NOT SEEM TO WORK PROPERLY.
      NBLOCK=((NSQ-1)/318 + 1)*(2*N + 3)
      CALL NEWLIN
      WRITE(IOFILE,87)NBLOCK
   87 FORMAT(I10,43H BLOCKS OF RANDOM DISC STORAGE ARE REQUIRED)
      NLINKS=(NBLOCK-1)/12 + 2
      CALL NEWLIN
      WRITE(IOFILE,86)NLINKS
```

```
86 FORMAT(I10,42H LINKS OF RANDOM DISC STORAGE ARE REQUIRED)
   NTIMES=0
81 CALL GETMOR(1,IERR,NLINKS,JFILE1)
   IF(IERR.EQ.0) GO TO 88
C     REQUEST WAS REFUSED
   CALL NEWLIN
   WRITE(IOFILE,82)
82 FORMAT(29H REQUEST FOR DISC WAS REFUSED)
   NTIMES=NTIMES+1
   IF(NTIMES.LT.MAXTIM) GO TO 81
   CALL NEWLIN
   WRITE(IOFILE,83) NTIMES
83 FORMAT(25H REQUEST FOR DISC REFUSED,I8,16H TIMES. GIVE UP.)
   STOP
88 CALL SETSIZ(NSQ)
C     SEE IF WE HAVE ENOUGH CORE
   MCORE=5*NSQ + N
   CALL NEWLIN
   WRITE(IOFILE,101)ICORE
101 FORMAT(I10,38H WORDS OF CORE ARE CURRENTLY AVAILABLE)
   CALL NEWLIN
   WRITE(IOFILE,102)MCORE
102 FORMAT(I10,39H WORDS OF CORE ARE REQUIRED FOR THE JOB)
   IF(MCORE.LE.ICORE) GO TO 3
C     GET MORE CORE
C     ICORE IS AUTOMATICALLY UPDATED TO REFLECT THE ACTUAL NUMBER
C      OF WORDS THERE ARE UPON RETURN
C     THE ROUTINE DOES NOT FAIL
C     IT KEEPS TRYING UNTIL IT GETS THE CORE IT WANTS
   CALL NEWLIN
   WRITE(IOFILE,103)
103 FORMAT(33H GET THE ADDITIONAL CORE REQUIRED)
   CALL GIMME(MCORE,ICORE,CORE)
C     COMPUTE LINEAR OFFSET FOR EACH MATRIX
 3 IOFF(1)=1
   DO 4 I=2,MAXOFF
 4 IOFF(I)=IOFF(I-1)+NSQ
   NUM=N
C     RESERVE SPACE FOR 2*N MATRICES
   J=2*NUM
   DO 91 I=1,J
91 CALL NEWNUM(K)
   CALL FIRST(NUM,CORE(IOFF(1)),NUM,NUM,CORE(IOFF(2)),NUM,NUM,
```

```
     1                    CORE(IOFF(3)),NUM,NUM,CORE(IOFF(4)),NUM,NUM,
     1                    CORE(ICFF(5)),NUM,NUM,CORE(IOFF(6)),NUM)
      CALL FINISH
      STOP
      END
CPIRST          FIRST
      SUBROUTINE FIRST(NUM,Q1,NR1,NC1,Q2,NR2,NC2,Q3,NR3,NC3,Q4,NR4,NC4,
     1                 Q5,NR5,NC5,IVEC,NVEC)
      COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
      COMMON /DEBUG/IDEBUG
      DIMENSION IVEC(NVEC)
      REAL Q1(NR1,NC1)
      REAL Q2(NR2,NC2)
      REAL Q3(NR3,NC3)
      REAL Q4(NR4,NC4)
      REAL Q5(NR5,NC5)
      DIMENSION IQ1(4)
      DIMENSION IQ2(4)
      DIMENSION IQ3(4)
      DIMENSION IQ4(4)
      DIMENSION IQ5(4)
      IQ1(1)=0
      IQ2(1)=0
      IQ3(1)=0
      IQ4(1)=0
      IQ5(1)=0
      IQ1(2)=0
      IQ2(2)=0
      IQ3(2)=0
      IQ4(2)=0
      IQ5(2)=0
      IQ1(3)=NR1
      IQ2(3)=NR2
      IQ3(3)=NR3
      IQ4(3)=NR4
      IQ5(3)=NR5
      IQ1(4)=NC1
      IQ2(4)=NC2
      IQ3(4)=NC3
      IQ4(4)=NC4
      IQ5(4)=NC5
      CALL SETUPA(Q1,NR1,NC1,Q2,NR2,NC2,Q3,NR3,NC3,Q4,NR4,NC4,
     1            Q5,NR5,NC5,IVEC,NVEC,NUM)
```

```
C
C
C     GENERATE RECURSION COEFFICIENTS A' AND B'
      CALL RSTR(Q1,IQ1,1)
      CALL RSTR(Q2,IQ2,2)
      CALL RSTR(Q3,IQ3,3)
      DO 10 NAME=3,(NUM*2-1),2
      CALL MMPY(Q3,IQ3,Q2,IQ2,Q4,IQ4,IERR)
      CALL MATIDN(Q5,IQ5,NUM)
      CALL MSUB(Q5,IQ5,Q4,IQ4,Q4,IQ4,IERR)
      CALL MINV(Q4,NUM,NUM,IVEC,DET)
      CALL RSTR(Q5,IQ5,NAME+1)
      CALL MMPY(Q4,IQ4,Q5,IQ5,Q2,IQ2,IERR)
      CALL SAVE(Q2,IQ2,NAME+1)
      CALL MMPY(Q3,IQ3,Q1,IQ1,Q5,IQ5,IERR)
      CALL MMPY(Q4,IQ4,Q5,IQ5,Q1,IQ1,IERR)
      CALL SAVE(Q1,IQ1,NAME)
   10 CONTINUE
      IF(IDEBUG.NE.0) GO TO 666
      CALL NEWLIN
      WRITE(IOFILE,566)
  566 FORMAT(14H DEBUG POINT C)
      CALL MATMAT
  666 CONTINUE
C
C
C     USE A' AND B' TO SOLVE FOR UNKNOWNS
      CALL MATZER(Q1,IQ1)
C     Q1=X
      CALL RSTR(Q2,IQ2,NUM*2-1)
      CALL MATZER(Q2,IQ2)
      CALL SAVE(Q2,IQ2,NUM*2-1)
C     Q2=T
      NAME=NUM*2
   20 NAME=NAME-2
      CALL RSTR(Q3,IQ3,NAME)
C     Q3= S,Q,O,...
      CALL MMPY(Q3,IQ3,Q2,IQ2,Q1,IQ1,IERR)
      IF(IERR.NE.0) CALL QUIT(IERR)
C     Q1= X
      CALL RSTR(Q2,IQ2,NAME-1)
C     Q2= R,P,N,...
      CALL MADD(Q1,IQ1,Q2,IQ2,Q2,IQ2,IERR)
```

```
      IF(IERR.NE.0) CALL QUIT(IERR)
      CALL SAVE(Q2,IQ2,NAME-1)
C     Q2= R,P,N,...
      IF(NAME.GE.4) GO TO 20
      IF(IDEBUG.NE.0) GO TO 667
      CALL NEWLIN
      WRITE(IOFILE,567)
  567 FORMAT(14H DEBUG POINT D)
      CALL MATMAT
  667 CONTINUE
C
C
C     PRINT OUT THE RESULTS
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,8801)
 8801 FORMAT(9H A MATRIX)
      CALL RSTR(Q1,IQ1,1)
      CALL PMAT(Q1,IQ1(1),IQ1(2),IQ1(3),IQ1(4))
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,8802)
 8802 FORMAT(9H C MATRIX)
      CALL RSTR(Q1,IQ1,3)
      CALL PMAT(Q1,IQ1(1),IQ1(2),IQ1(3),IQ1(4))
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,8803)
 8803 FORMAT(9H P MATRIX)
      CALL RSTR(Q1,IQ1,(NUM-2)*2-1)
      CALL PMAT(Q1,IQ1(1),IQ1(2),IQ1(3),IQ1(4))
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,8804)
 8804 FORMAT(9H R MATRIX)
      CALL RSTR(Q1,IQ1,(NUM-1)*2-1)
      CALL PMAT(Q1,IQ1(1),IQ1(2),IQ1(3),IQ1(4))
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,8805)
 8805 FORMAT(9H T MATRIX)
      CALL RSTR(Q1,IQ1,(NUM  )*2-1)
      CALL PMAT(Q1,IQ1(1),IQ1(2),IQ1(3),IQ1(4))
```

```
      RETURN
C     END FIRST
      END
CSETUPA          SETUPA
      SUBROUTINE SETUPA(H,NRH,NCH,V,NRV,NCV,C,NRC,NCC,X,NRX,NCX,
     1                  Y,NRY,NCY,IVEC,NVEC,NUM)
      COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
      COMMON /DEBUG/IDEBUG
      DIMENSION IVEC(NVEC)
      DIMENSION H(NRH,NCH)
      DIMENSION V(NRV,NCV)
      DIMENSION C(NRC,NCC)
      DIMENSION X(NRX,NCX)
      DIMENSION Y(NRY,NCY)
      DIMENSION IH(4)
      DIMENSION IV(4)
      DIMENSION IC(4)
      DIMENSION IX(4)
      DIMENSION IY(4)
      DIMENSION KARD(14)
      DATA IHEAT/4HHEAT/
      IH(1)=NUM
      IH(2)=NUM
      IH(3)=NRH
      IH(4)=NCH
      IV(1)=NUM
      IV(2)=NUM
      IV(3)=NRV
      IV(4)=NCV
      IC(1)=NUM
      IC(2)=1
      IC(3)=NRC
      IC(4)=NCC
      IX(1)=NUM
      IX(2)=NUM
      IX(3)=NRX
      IX(4)=NCX
      IY(1)=NUM
      IY(2)=NUM
      IY(3)=NRY
      IY(4)=NCY
      CALL NEWNUM(INDEXC)
      CALL NEWNUM(INDEXH)
```

```
      CALL NEWNUM(INDEXV)
      CALL MATZER(H,IH)
      CALL MATZER(V,IV)
      CALL MATZER(C,IC)
      PI=3.14159265
      C1=.5/PI
      R1=C1+(ALOG(2.0))/(2.0*PI)
      R2=R1/2.0
      H(1,1)=1.0/R1
      H(NUM,1)=1.0/R1
      DO 180 M=2,NUM-1
  180 H(M,1)=2.0*H(1,1)
      DO 210 M=1,NUM
  210 V(M,1)=PI/2.0
      DO 310 M=1,NUM
      DO 300 N=2,NUM
      H(M,N)=1.0/(1.0/(4.0*PI)*ALOG(FLOAT(N)/(FLOAT(N)-1.0)))
      V(M,N)=PI*4.0*(FLOAT(N)-1.0)
      IF(M.GT.1) GO TO 280
      H(1,N)=.5*H(1,N)
  280 H(NUM,N)=.5*H(NUM,N)
      V(M,10)=PI*(4.0* FLOAT(N)-5.0)/2.0
  300 CONTINUE
  310 CONTINUE
      DO 340 M=1,NUM
  340 H(M,NUM)=0.0
      DO 370 N=1,NUM
  370 V(1,N)=0.0
      C2=1.0
      C3=1.0
      DO 460 M=1,NUM
      DO 450 N=1,NUM
      V(M,N)=C3*V(M,N)
      H(M,N)=C2*H(M,N)
  450 CONTINUE
  460 CONTINUE
      READ(INFILE,461,END=468)KARD
  461 FORMAT(13A6,A2)
      CALL NEWLIN
      WRITE(IOFILE,464)KARD
  464 FORMAT(2H *,13A6,A2,1H*)
      IF(KARD(1).EQ.IHEAT) GO TO 462
      CALL NEWLIN
```

```
      WRITE(IOFILE,463)
  463 FORMAT(35H ABOVE CARD SHOULD BE A 'HEAT' CARD)
      STOP
  462 DECODE(KARD,465)I1,I2
  465 FORMAT(10X,2I5)
      IF(I1.LE.0.OR.I1.GT.I2 ) GO TO 472
      IF(I2.GT.NUM) GO TO 473
      GO TO 479
  472 CALL NEWLIN
      WRITE(IOFILE,474)
  474 FORMAT(20H I1 IS OUT OF BOUNDS)
      STOP
  473 CALL NEWLIN
      WRITE(IOFILE,475)
  475 FORMAT(20H I2 IS OUT OF BOUNDS)
      STOP
  479 CONTINUE
      DO 466 N=1,NUM
  466 C(N,1)=0.0
      DO 467 N=I1,I2
  467 C(N,1)=V(2,N)*2.0
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,469)
  469 FORMAT(26H INITIAL HEAT INPUT MATRIX)
      CALL PMAT(C,IC(1),IC(2),IC(3),IC(4))
      GO TO 471
  468 CALL NEWLIN
  470 FORMAT(49H UNEXPECTED END OF FILE.  EXPECTING A 'HEAT' CARD)
      WRITE(IOFILE,470)
      STOP
  471 CONTINUE
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,701)
  701 FORMAT(9H H MATRIX)
      CALL PMAT(H,IH(1),IH(2),IH(3),IH(4))
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,702)
  702 FORMAT(9H V MATRIX)
      CALL PMAT(V,IV(1),IV(2),IV(3),IV(4))
      CALL SAVE(C,IC,INDEXC)
```

```
      CALL SAVE(H,IH,INDEXH)
      CALL SAVE(V,IV,INDEXV)
      IF(IDEBUG.NE.0) GO TO 664
      CALL NEWLIN
      WRITE(IOFILE,564)
  564 FORMAT(14H DEBUG POINT A)
      CALL MATMAT
  664 CONTINUE
C
C     THE MATRICES ON THE DIAGONAL (AND OF COURSE THEIR INVERSES)
C     ARE DIFFERENT ONLY FOR THE FIRST, SECOND, AND LAST TIMES.
      DO 1150 M=1,NUM
      IGO=1
      IF(M.EQ.1.OR.M.EQ.2.OR.M.EQ.NUM)IGO=0
      IX(1)=NUM
      IX(2)=NUM
      CALL MATZER(X,IX)
      IY(1)=NUM
      IY(2)=NUM
      CALL MATZER(Y,IY)
      IF(IGO.EQ.0)IC(1)=NUM
      IF(IGO.EQ.0)IC(2)=NUM
      IF(IGO.EQ.0)CALL MATZER(C,IC)
      CALL RSTR(H,IH,INDEXH)
      DO 720 N=1,NUM
      IF(N.EQ.1.OR.N.EQ.NUM) GO TO 590
      IF(IGO.EQ.0)C(N,N-1)=0.0
      IF(IGO.EQ.0)C(N,N+1)=0.0
  590 IF(N.LT.2) GO TO 610
      IF(IGO.EQ.0)C(N,N-1)=-H(M,N-1)
  610 IF(N.GE.NUM) GO TO 630
      IF(IGO.EQ.0)C(N,N+1)=-H(M,N)
  630 IF(M.GE.NUM) GO TO 640
      X(N,N)=-V(M+1,N)
  640 Y(N,N)=-V(M,N)
      C33=0.0
      IF(N.EQ.1) GO TO 680
      C33=C(N,N-1)
  680 IF(N.GE.NUM) GO TO 700
      C33=C33+C(N,N+1)
  700 IF(IGO.EQ.0)C(N,N)=-(C33+X(N,N)+Y(N,N))
  720 CONTINUE
      IF(IGO.EQ.0)CALL MINV(C,NUM,NUM,IVEC,DET)
```

```
      CALL MMPY(C,IC,X,IX,H,IH,IERR)
      IF(IERR.NE.0) CALL QUIT(IERR)
      CALL MNEG(H,IH)
C     SAVE H (B,D,F,...)
      CALL SAVE(H,IH,M*2)
      IF(M.GT.1) GO TO 888
      CALL RSTR(H,IH,INDEXC)
      CALL MMPY(C,IC,H,IH,X,IX,IERR)
      IF(IERR.GT.0) CALL QUIT(IERR)
C     SAVE A
      CALL SAVE(X,IX,M*2-1)
      GO TO 1150
  888 CALL MMPY(C,IC,Y,IY,X,IX,IERR)
      CALL MNEG(X,IX)
      IF(IERR.NE.0) CALL QUIT(IERR)
C     SAVE (C,E,G,...)
      CALL SAVE(X,IX,M*2-1)
 1150 CONTINUE
      IF(IDEBUG.NE.0) GO TO 665
      CALL NEWLIN
      WRITE(IOFILE,565)
  565 FORMAT(14H DEBUG POINT 8)
      CALL MATMAT
  665 CONTINUE
      RETURN
C     END SETUPA
      END
CSAVE          SAVE
      SUBROUTINE SAVE(A,IA,INDEX)
      COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
      DIMENSION A(1)
      DIMENSION IA(4)
      DIMENSION IB(3)
      DATA NRW/0/
      DATA KOUNT/0/
      IF(INDEX.LT.1.OR.INDEX.GT.KOUNT) GO TO 801
      IF(IA(1).LE.0) CALL QUIT(1)
      IF(IA(2).LE.0) CALL QUIT(1)
C     CHECK TO SEE IF RANDOM RECORD SIZE IS LARGE ENOUGH
      IF((IA(1)*IA(2)).GT.MAXWRD) CALL QUIT(1)
      IB(1)=IA(1)
      IB(2)=IA(2)
      IB(3)=1
```

```
      WRITE(IFILE2'INDEX)IB
      NR=IA(1)
      NC=IA(2)
      NCOLS=IA(4)
      I2=NR-1
      I3=NCOLS*(NR-1)+1
      WRITE(IFILE1'INDEX)((A(I),I=I1,I1+I2),I1=1,I3,NCOLS)
      NRW=NRW+1
      RETURN
C
      ENTRY RSTR(A,IA,INDEX)
      IF(INDEX.LT.1.OR.INDEX.GT.KOUNT) GO TO 801
       READ(IFILE2'INDEX)IB
      IF(IB(3).EQ.0) CALL QUIT(INDEX)
C     CHECK TO SEE IF THE DIMENSIONS OF THE MATRIX ARE LARGE ENOUGH
      IF(IB(1).GT.IA(3)) CALL QUIT(1)
      IF(IB(2).GT.IA(4)) CALL QUIT(1)
      IA(1)=IB(1)
      IA(2)=IB(2)
      NR=IA(1)
      NC=IA(2)
      NCOLS=IA(4)
      I2=NR-1
      I3=NCOLS*(NR-1)+1
       READ(IFILE1'INDEX)((A(I),I=I1,I1+I2),I1=1,I3,NCOLS)
      NRW=NRW+1
      RETURN
C
      ENTRY FINISH
      CALL NEWPAG
      CALL NEWLIN
      WRITE(IOFILE,1111)NRW
 1111 FORMAT(I10,21H READ-WRITES EXECUTED)
      CALL NEWLIN
      WRITE(IOFILE,6)
    6 FORMAT(12H NORMAL HALT)
      STOP
C
      ENTRY SETSIZ(NUMBR)
      MAXWRD=NUMBR
      CALL RANSIZ(IFILE1,MAXWRD,0)
      CALL RANSIZ(IFILE2,3)
      RETURN
```

```
C
      ENTRY SETNUM(NUMBR)
      MAXKNT=NUMBR
      CALL SETDUM(NUMBR)
      RETURN
C
      ENTRY NEWNUM(NUMBR)
      KOUNT=KOUNT+1
      IF(KOUNT.LE.MAXKNT) GO TO 501
      WRITE(IOFILE,502)MAXKNT
  502 FORMAT(10H MORE THAN,I10, 9H MATRICES)
      CALL QUIT(1)
  501 CONTINUE
      NUMBR=KOUNT
      IB(1)=0
      IB(2)=0
      IB(3)=0
      WRITE(IFILE2'KOUNT)IB
      RETURN
C
      ENTRY MATZER(A,IA)
      NSQ=IA(3)*IA(4)
      DO 1 I=1,NSQ
    1 A(I)=0.0
      RETURN
C
C     TO NEGATE A MATRIX
      ENTRY MNEG(A,IA)
      NSQ=IA(3)*IA(4)
      DO 82 I=1,NSQ
   82 A(I)=-A(I)
      RETURN
C
      ENTRY MATIDN(A,IA,NUM)
      IF(IA(3).LT.NUM.OR.IA(4).LT.NUM) CALL QUIT(1)
      IA(1)=NUM
      IA(2)=NUM
      NSQ=IA(3)*IA(4)
      DO 2 I=1,NSQ
    2 A(I)=0.0
      DO 3 I=1,NUM
      NSQ=IA(3)*(I-1) + I
    3 A(NSQ)=1.0
```

```
      RETURN
C
  801 CALL NEWLIN
      WRITE(IOFILE,802)INDEX
  802 FORMAT(20H BAD MATRIX INDEX OF,I10)
      CALL QUIT(1)
      STOP
C     END SAVE
      END
CNEWLIN          NEWLIN
      SUBROUTINE NEWLIN
      COMMON /DATTIM/ITIME(2),IDATE(2)
      PARAMETER MAX=20
      DIMENSION IHEAD(MAX)
      DIMENSION JHEAD(MAX)
      DIMENSION IVEC(NWORDS)
      DATA JHEAD/MAX*6H      /
      DATA IOFILE/6/
      DATA MAXLIN/55/
      DATA INAME/1H /
      DATA NLINES/0/
      DATA IG01/1/
      DATA IPAGE/0/
      DATA N2/3/
      DATA IBLANK/1H /
      IG02=1
      GO TO ( 1,32),IG01
   32 IF(NLINES.GT.0) GO TO 5
      IPAGE=IPAGE+1
      WRITE(IOFILE,18)
   18 FORMAT(1H1,//)
      WRITE(IOFILE,6) ITIME,      IDATE,IPAGE
    6 FORMAT( 9H TIME IS ,2A6 , 4H ON ,          2A6,80X,
    1 5HPAGE=,I5)
      WRITE(IOFILE,87)JHEAD
   87 FORMAT(10H    TITLE=,20A6)
      WRITE(IOFILE,7) IHEAD
    7 FORMAT(10H SUBTITLE=,20A6)
      DO 8 I=1,N2
    8 WRITE(IOFILE,9)
    9 FORMAT(1H )
      NLINES=N2+3
    5 NLINES=NLINES+1
```

```
      IF(NLINES.GT.MAXLIN)NLINES=0
      RETURN
C
      ENTRY NEWPAG
      NLINES=0
      RETURN
C
      ENTRY SETHED(NWORDS,IVEC)
      IGO2=2
      GO TO (1,2),IGO1
    1 CALL TIMDAT( ITIME,IDATE)
      IGO1=2
    2 NLINES=0
      IPAGE=0
      DO 3 I=1,MAX
    3 IHEAD(I)=IBLANK
      GO TO (32,22),IGO2
   22 DO 4 I=1,NWORDS
    4 IHEAD(I)=IVEC(I)
      RETURN
C
      ENTRY SUPERT(NWORDS,IVEC)
      DO 80 I=1,NWORDS
   80 JHEAD(I)=IVEC(I)
      RETURN
C     END NEWLIN
      END
CQUIT           QUIT
      SUBROUTINE QUIT(IERR)
      COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
      DIMENSION IB(3)
      NSQ=0
      NSQ=NSQ**NSQ
      WRITE(IOFILE,21)IERR
   21 FORMAT(22H QUIT BECAUSE OF IERR=,I10)
      IGO=1
      GO TO 777
C
      ENTRY MATMAT
      IGO=0
  777 CONTINUE
      WRITE(IOFILE,12)
   12 FORMAT(19H MATRIX INFORMATION)
```

```
      WRITE(IOFILE,13)
   13 FORMAT(80H         MATRIX NUMBER      NO. ROWS IN USE      NO. COLS IN
     1 USE          EVER USED)
      DO 10 I=1,MAXKNT
      READ(IFILE2'I)IB
   10 WRITE(IOFILE,11)I,IB
   11 FORMAT(4I20)
      IF(IGO.EQ.0) RETURN
      CALL PDUMP
      STOP
C
      ENTRY SETDUM(IERR)
      MAXKNT=IERR
      RETURN
C     END QUIT
      END
*INVRS          SUBROUTINE TO OBTAIN INVERSE OF MATRIX, CALL DECOM FIRST INVRS020
*                              CD600D4.007        DATE 05/04/65          INVRS030
      SUBROUTINE INVRS(A,INTR,MSIZE,NN)                                  INVRS040
      DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)                               INVRS050
*     OBTAINS EXPLICIT INVERSE OF DECOMPOSED MATRIX                      INVRS060
*     SUBROUTINE DECOM MUST BE CALLED FIRST                              INVRS070
      N=NN                                                               INVRS080
      IF(INTR(N))18,17,18                                                INVRS090
    1 DO 13 K=1,N                                                        INVRS100
      KM=K-1                                                             INVRS110
      IF(KM)2,7,2                                                        INVRS120
    2 IF(K-N)3,7,3                                                       INVRS130
*     COMPLETE REDUCTION BELOW DIAGONAL                                  INVRS140
    3 KP=K+1                                                             INVRS150
      DO 6 I=KP,N                                                        INVRS160
      X=A(I,K)                                                           INVRS170
      IF(X)4,6,4                                                         INVRS180
    4 DO 5 J=1,KM                                                        INVRS190
    5 A(I,J)=A(K,J)*X+A(I,J)                                             INVRS200
    6 CONTINUE                                                           INVRS210
*     DIVIDE THROUGH BY PIVOT ELEMENT                                    INVRS220
    7 X=1.0/A(K,K)                                                       INVRS230
      A(K,K)=1.0                                                         INVRS240
      DO 8 J=1,N                                                         INVRS250
    8 A(K,J)=A(K,J)*X                                                    INVRS260
      IF(KM)9,13,9                                                       INVRS270
*     REDUCE TERMS ABOVE DIAGONAL                                        INVRS280
```

```
    9 DO 12 I=1,KM                                                      INVRS290
      X=-A(I,K)                                                         INVRS300
      IF(X)10,12,10                                                     INVRS310
   10 A(I,K)=0.0                                                        INVRS320
      DO 11 J=1,N                                                       INVRS330
   11 A(I,J)=A(K,J)*X+A(I,J)                                            INVRS340
   12 CONTINUE                                                          INVRS350
   13 CONTINUE                                                          INVRS360
*     INTERCHANGE COLUMNS                                               INVRS370
*     KM=N-1 FROM PREVIOUS LOOP                                         INVRS380
      DO 16 J=1,KM                                                      INVRS390
      K=N-J                                                             INVRS400
      KP=INTR(K)                                                        INVRS410
      IF(KP)14,16,14                                                    INVRS420
   14 DO 15 I=1,N                                                       INVRS430
      X=A(I,KP)                                                         INVRS440
      A(I,KP)=A(I,K)                                                    INVRS450
   15 A(I,K)=X                                                          INVRS460
   16 CONTINUE                                                          INVRS470
   17 RETURN                                                            INVRS480
   18 KM=N-1                                                            INVRS490
      DO 21 I=2,KM                                                      INVRS500
      K=INTR(I)                                                         INVRS510
      IF(K)19,21,19                                                     INVRS520
   19 KP=I-1                                                            INVRS530
      DO 20 J=1,KP                                                      INVRS540
      X=A(I,J)                                                          INVRS550
      A(I,J)=A(K,J)                                                     INVRS560
   20 A(K,J)=X                                                          INVRS570
   21 CONTINUE                                                          INVRS580
      GO TO 1                                                           INVRS590
      END                                                               INVRS600
*MINV                            MATRIX INVERSE ROUTINE                 MINV0020
*                                CD600D4.007         DATE 05/04/65      MINV0030
      SUBROUTINE MINV(A,MSIZE,N,INTR,DET)                               MINV0040
      DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)                              MINV0050
      CALL DECOM(A,INTR,MSIZE,N)                                        MINV0060
      CALL DTMN(A,INTR,MSIZE,N,DET)                                     MINV0070
      CALL INVRS(A,INTR,MSIZE,N)                                        MINV0080
      RETURN                                                            MINV0130
      END                                                               MINV0140
*MMPY            CD600D4.012      MATRIX MULTIPLY ROUTINE 05/18/66      MMPY0002
*                COPYRIGHT 1966 BY GENERAL ELECTRIC COMPANY             MMPY0003
```

```
      SUBROUTINE MMPY(A,IDA,B,IDB,C,IDC,IND)                            MMPY0040
      DIMENSION A(1),B(1),C(1),IDA(4),IDB(4),IDC(4)                     MMPY0050
      IA=IDA(1)                                                         MMPY0060
      JA=IDA(2)                                                         MMPY0070
      MA=IDA(3)                                                         MMPY0080
      IB=IDB(1)                                                         MMPY0090
      JB=IDB(2)                                                         MMPY0100
      MB=IDB(3)                                                         MMPY0110
      IC=IDC(1)                                                         MMPY0120
      JC=IDC(2)                                                         MMPY0130
      MC=IDC(3)                                                         MMPY0140
      NC=IDC(4)                                                         MMPY0150
      IND=0                                                             MMPY0160
      IF(JA.NE.IB)IND=2
      IF((MC.LT.IA).OR.(NC.LT.JB))IND=1                                 MMPY0180
      IF(IND.NE.0)RETURN                                                MMPY0190
      IDC(1)=IDA(1)
      IDC(2)=IDB(2)
      DO 1 I=1,IA                                                       MMPY0200
      DO 1 K=1,JB
      LC=MC*(K-1)+I                                                     MMPY0220
      C(LC)=0.0                                                         MMPY0230
      KB=MB*(K-1)                                                       MMPY0240
      DO 1 J=1,IB
      LA=MA*(J-1)+I                                                     MMPY0260
      LB=KB+J                                                           MMPY0270
    1 C(LC)=C(LC)+A(LA)*B(LB)                                           MMPY0280
      RETURN                                                            MMPY0290
      END                                                               MMPY0300
*MADD          CD60004.010 MATRIX ADD ROUTINE 05/18/66                  MADD0002
*              COPYRIGHT 1966 BY GENERAL ELECTRIC COMPANY               MADD0003
      SUBROUTINE MADD(A,IDA,B,IDB,C,IDC,IND)                            MADD0040
      DIMENSION A(1),B(1),C(1),IDA(4),IDB(4),IDC(4)                     MADD0050
      IA=IDA(1)                                                         MADD0060
      JA=IDA(2)                                                         MADD0070
      IB=IDB(1)                                                         MADD0080
      JB=IDB(2)                                                         MADD0090
      IC=IDC(1)                                                         MADD0093
      JC=IDC(2)                                                         MADD0096
      MA=IDA(3)                                                         MADD0100
      MB=IDB(3)                                                         MADD0110
      MC=IDC(3)                                                         MADD0120
      NC=IDC(4)                                                         MADD0130
```

```
      IND=0
      IF((IA.NE.IB).OR.(JA.NE.JB))IND=2                                 MADD0140
      IF((MC.LT.IA).OR.(NC.LT.JA))IND=1                                 MADD0160
      IF(IND.NE.0)RETURN                                                MADD0170
      IDC(1)=IDA(1)
      IDC(2)=IDA(2)
      DO 1 J=1,JA                                                       MADD0180
      JM=J-1                                                            MADD0190
      LA=MA*JM                                                          MADD0200
      LB=MB*JM                                                          MADD0210
      LC=MC*JM                                                          MADD0220
      DO 1 I=1,IA                                                       MADD0230
      LA=LA+1                                                           MADD0240
      LB=LB+1                                                           MADD0250
      LC=LC+1                                                           MADD0260
    1 C(LC)=A(LA)+B(LB)                                                 MADD0270
      RETURN                                                            MADD0280
      END                                                               MADD0290
*MSUB          CD60004.011     MATRIX SUBTRACT ROUTINE 05/18/66         MSUB0002
*              COPYRIGHT 1966 BY GENERAL ELECTRIC COMPANY               MSUB0003
      SUBROUTINE MSUB(A,IDA,B,IDB,C,IDC,IND)                            MSUB0040
      DIMENSION A(1),B(1),C(1),IDA(4),IDB(4),IDC(4)                     MSUB0050
      IA=IDA(1)                                                         MSUB0060
      JA=IDA(2)                                                         MSUB0070
      IB=IDB(1)                                                         MSUB0080
      JB=IDB(2)                                                         MSUB0090
      IC=IDC(1)                                                         MSUB0093
      JC=IDC(2)                                                         MSUB0096
      MA=IDA(3)                                                         MSUB0100
      MB=IDB(3)                                                         MSUB0110
      MC=IDC(3)                                                         MSUB0120
      NC=IDC(4)                                                         MSUB0130
      IND=0                                                             MSUB0140
      IF((IA.NE.IB).OR.(JA.NE.JB))IND=2
      IF((MC.LT.IA).OR.(NC.LT.JA))IND=1                                 MSUB0160
      IF(IND.NE.0)RETURN                                                MSUB0170
      IDC(1)=IDA(1)
      IDC(2)=IDA(2)
      DO 1 J=1,JA                                                       MSUB0180
      JM=J-1                                                            MSUB0190
      LA=MA*JM                                                          MSUB0200
      LB=MB*JM                                                          MSUB0210
      LC=MC*JM                                                          MSUB0220
```

```
      DO 1 I=1,IA
      LA=LA+1
      LB=LB+1
      LC=LC+1
    1 C(LC)=A(LA)-B(LB)
      RETURN
      END
*DECOM        SUBROUTINE TO DECOMPOSE MATRIX FOR SIMULTANEOUS EQUATIONS
*                            CD600D4,007         DATE 05/04/65
      SUBROUTINE DECOM(A,INTR,MSIZE,NN)
      DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)
*     MATRIX DECOMPOSITION USED WITH SOLV SUBROUTINE FOR SOLUTION
*      OF LINEAR SYSTEMS
*     IF MATRIX A IS SINGULAR INTR(N) WILL BE SET TO ZERO
      N=NN
      NTR=1
      NM=N-1
      DO 10 J=1,NM
      AMAX=ABS(A(J,J))
      JP=J+1
      IN=0
      DO 2 I=JP,N
      AT=ABS(A(I,J))
      IF(AMAX-AT)1,2,2
    1 AMAX=AT
      IN=I
    2 CONTINUE
      IF(AMAX)4,3,4
    3 INTR(J)=J
      GO TO 11
    4 IF(IN)5,7,5
    5 NTR=-NTR
      DO 6 I=J,N
      AT=A(J,I)
      A(J,I)=A(IN,I)
    6 A(IN,I)=AT
    7 INTR(J)=IN
      AMAX=-1.0/A(J,J)
      DO 10 I=JP,N
      IF(A(I,J))8,10,8
    8 AT=A(I,J)*AMAX
      A(I,J)=AT
      DO 9 K=JP,N
```

```
    9 A(I,K)=A(J,K)*AT+A(I,K)                                   DECOM380
   10 CONTINUE                                                  DECOM390
      IF(A(N,N))12,11,12                                        DECOM400
   11 NTR=0                                                     DECOM410
   12 INTR(N)=NTR                                               DECOM420
      RETURN                                                    DECOM430
      END                                                       DECOM440
*DTMN                     DETERMINANT EVALUATION SUBROUTINE     DTMN0020
*                         CD600D4.007       DATE 05/04/65       DTMN0030
      SUBROUTINE DTMN(A,INTR,MSIZE,NN,DET)                      DTMN0040
      DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)                      DTMN0050
*     COMPUTES DET, THE DETERMINANT OF THE DECOMPOSED MATRIX    DTMN0060
*     SUBROUTINE DECOM MUST BE CALLED FIRST                     DTMN0070
*     INTR(N) WILL CONTAIN INTEGER POWER OF TEN OF MULTIPLYING FACTOR DTMN0080
      EP=1.E38                                                  DTMN0090
      NE=38                                                     DTMN0100
      N=NN                                                      DTMN0110
      NTR=INTR(N)                                               DTMN0120
      IF(NTR)2,1,3                                              DTMN0130
    1 DTT=0.0                                                   DTMN0140
      GO TO 10                                                  DTMN0150
    2 DTT=-10.                                                  DTMN0160
      GO TO 4                                                   DTMN0170
    3 DTT=0.1                                                   DTMN0180
    4 DO 9 I=1,N                                                DTMN0190
      DT=ABS(DTT)                                               DTMN0200
      IF(ABS(A(I,I))-1.0)5,9,7                                  DTMN0210
    5 IF(DT-1.)6,9,9                                            DTMN0220
    6 DTT=DTT*EP                                                DTMN0230
      NTR=NTR-NE                                                DTMN0240
      GO TO 9                                                   DTMN0250
    7 IF(DT-1.0)9,9,8                                           DTMN0260
    8 DTT=DTT/EP                                                DTMN0270
      NTR=NTR+NE                                                DTMN0280
    9 DTT=DTT*A(I,I)                                            DTMN0290
      INTR(N)=NTR                                               DTMN0300
   10 DET=DTT                                                   DTMN0310
      RETURN                                                    DTMN0320
      END                                                       DTMN0330
*PMAT                     SUBROUTINE TO PRINT MATRIX            PMAT0020
      SUBROUTINE PMAT(A,NR,NC,MM,NN)
      DIMENSION A(MM,NN),P(6)                                   PMAT0050
      NROW=NR                                                   PMAT0060
```

```
      NCOL=NC
      I=1                                                               PMAT0070
      J=1                                                               PMAT0080
    1 IP=I                                                              PMAT0090
      JP=J                                                              PMAT0100
      DO 2 K=1,6                                                        PMAT0110
      KK=K                                                              PMAT0120
      P(K)=A(I,J)                                                       PMAT0130
      J=J+1                                                             PMAT0140
      IF(J.GT.NCOL)GO TO 3                                              PMAT0150
    2 CONTINUE                                                          PMAT0160
      CALL NEWLIN                                                       PMAT0170
      WRITE(6,4)IP,JP,(P(K),K=1,KK)
    4 FORMAT(2I4,6(1PE16.8))                                            PMAT0180
      GO TO 1                                                           PMAT0190
    3 CALL NEWLIN                                                       PMAT0200
      WRITE(6,4)IP,JP,(P(K),K=1,KK)
      I=I+1
      J=1                                                               PMAT0220
      IF(I.LE.NROW)GO TO 1                                              PMAT0230
      RETURN                                                            PMAT0240
      END                                                               PMAT0250
$       GMAP      DECK                                                  PMAT0260
$       INCODE    IBMF                                                  GETMOR
        LBL       GETMOR,GETMOR
        TTL       GETMOR
        SYMDEF    GETMOR
        REM       TO OBTAIN MORE CORE OR DISC
        REM       CALL GETMOR(TYPE,RESULT,NUM,FC)
        REM       TYPE=0 FOR CORE
        REM       TYPE=1 FOR RANDOM DISC
        REM       TYPE=2 FOR LINKED DISC
        REM       RESULT=0 IF SUCCESSFUL
        REM       RESULT=1 IF UNSUCCESSFUL
        REM       NUM IS NUMBER OF LINKS DESIRED OR
        REM                NUMBER OF K (1024 WORDS) DESIRED
        REM       FC IS THE FILE CODE IN THE FORM 6H0000FC
        REM            (USED ONLY WHEN GETTING MORE DISC)
GETMOR  LDA       4,1*
        LDQ       2,1*
        SBQ       =6H000001
        TMI       STORA                  CORE REQUEST
        TZE       RANDOM                 RANDOM DISC REQUEST
```

```
         LDQ     5,1*                LINKED DISC REQUEST
         TRA     ORTOA
RANDOM   LDQ     5,1*
         ORQ     =1,DU
ORTOA    ORA     =2,DU
STORA    STA     ZERO
         MME     GEMORE
ZERO     BSS     1
         TRA     NO
YES      LDQ     =6H000000
         TRA     CONT
NO       LDQ     =6H000001
CONT     STQ     3,1*
         TRA     0,1
         END
$        GMAP    DECK,COMDK                              81524     090672GIMME
$        INCODE  IBMF
         TTL     GIVE-ME-MORE-CORE                                    IMME0001
         LBL     GIMME,GIVE-ME-MORE-CORE                              IMME0002
*                                                                     IMME0003
*  SUBROUTINE TO ADJUST SIZE OF SPECIFIED ARRAY TO SPECIFIED          IMME0004
*    NEW SIZE.  MME GEMORE IS USED TO ADD MORE WORDS OF MEMORY.       IMME0005
*    MME GEMREL IS USED TO RELEASE SURPLUS WORDS OF MEMORY.           IMME0006
*                                                                     IMME0007
*  CALLING SEQUENCE                                                   IMME0008
*    CALL GIMME ( NEWSIZ, OLDSIZ, ARRAY NAME )                        IMME0009
*                                                                     IMME0010
*  THE ARRAY SPECIFIED MUST BE LOADED AT THE VERY TOP OF MEMORY,      IMME0011
*    USING EITHER A  $ USE  CONTROL CARD OR A BLOCK DATA SUBPROGRAM.  IMME0012
*    ( IT IS ALSO NECESSARY THAT THE ARRAY BE IN LABELED COMMON. )    IMME0013
*                                                                     IMME0014
*  WARNING--THIS SUBROUTINE CHANGES THE SECOND ARGUMENT               IMME0015
*          TO REFLECT THE NEW SIZE OF THE ARRAY                       IMME0016
*                                                                     IMME0017
        SYMREF  .FCNV.
GIMME   SAVE                                                          IMME0018
*                                                                     IMME0019
*  THESE NEXT INSTRUCTIONS ARE USED TO MONITOR PERFORMANCE OF GIMME   IMME0020
*  THEY MAY BE REMOVED IF NOT WANTED,  ALONG WITH THE CODE AT SYMBOL  IMME0021
*  DONE.  SET DONE  EQU  GIMME+1 IF REMOVED.                          IMME0022
*                                                                     IMME0023
        MME     GETIME             GET TIME OF DAY                    IMME0024
        STQ     TEMP2+1                                               IMME0025
```

```
         MME     GELAPS            GET PROCESSOR TIME USED TILL NOW          IMME0026
         STQ     TEMP2                                                       IMME0027
*                                                                            IMME0028
*  CHECK THAT THE REFERENCED ARRAY IS LOADED AT THE TOP OF CORE              IMME0029
*                                                                            IMME0030
         SBAR    *+1               GET THE ADDRESS OF THE TOP OF CORE        IMME0031
         LDQ     **,DL                                                       IMME0032
         ANQ     =0777,DL                                                    IMME0033
         QLS     9
         EAX0    4,1*              GET ADDRESS OF ARRAY
         STX0    ARADR             SAVE
ARADR    SBQ     **,DL             SUBTRACT ADDRESS OF ARRAY
         LDA     LOWLOD            CHECK LOWLOAD
         CANA    =1B18,DL            INDICATOR
         TNZ     LOW               ...LOWLOADED
*
*  HIGHLOAD PRE-PROCESS AND CHECKS
*
         LDA     -1,DU             SET FLAG
         STCA    RLSFLG,70           TO ALLOW MEMORY RELEASE
         STQ     TEMP              SAVE ARRAY SIZE
         SBQ     3,1*              SUBTRACT CLAIMED ARRAY SIZE
         TZE     *+3               ...EQUAL
         CMPQ    1,DU              MAYBE ARRAY WAS LOADED ON ODD LOCATION    IMME0039
         TNZ     ERROR             ... SORRY, SOMETHING ELSE AT TOP OF CORE  IMME0040
         LDQ     TEMP              RESTORE ARRAY SIZE
         TRA     GIM               ...CONTINUE
*
*  LOWLOAD PREPROCESS AND CHECKS
*
LOW      LDX0    ARADR             ADDRESS OF ARRAY
         CMPX0   LIMITS            COMPARE WITH LOWEST UNUSED
         TMI     ERROR             ...ARRAY NOT LOADED ABOVE PROGRAM
         LXL0    LIMITS            ADDRESS OF HIGH UNUSED LIMIT
         CMPX0   ARADR             COMPARE WITH BASE OF ARRAY
         TMI     *+4               ...OK, LIMIT IS BELOW BASE
         LDX0    ARADR             ADJUST LIMIT
         SBLX0   =1,DU               TO BE
         SXL0    LIMITS                BELOW ARRAY
         LDA     2,1*              COMPUTE FLAG
         SBA     3,1*                INDICATING THAT
         STCA    RLSFLG,70             RELEASE MAY OCCUR (IF NEGATIVE)
GIM      STQ     3,1*              MAKE SURE OLD ARRAY SIZE IS CORRECT
```

```
*
*  COMPUTE THE NUMBER OF 1024 WORD BLOCKS THAT ARE REQUIRED                IMME0041
*                                                                          IMME0042
                                                                           IMME0043
       LDQ    2,1*          NUMBER OF WORDS REQUIRED                       IMME0044
       SBQ    3,1*          NUMBER OF WORDS ALREADY IN THE ARRAY           IMME0045
       ADQ    =1023,DL      ROUNDING UP FACTOR                             IMME0049
       QRS    10            DIVIDE BY 1024                                 IMME0050
       STQ    TEMP          SAVE NUMBER OF K TO GROW
       TZE    DONE          ...NO CORE ADJUSTMENT REQUIRED
*  IF GIMME IS NOT TO RELEASE CORE, INSERT THE FOLLOWING INSTRUCTION
*  HERE
****   TMI    DONE          ...NO MORE CORE REQUIRED
       LDA    TEMP          UPDATE THE NUMBER
       ALS    10              OF WORDS IN
       ASA    3,1*              THE ARRAY
*  IF GIMME IS NOT TO RELEASE SURPLUS CORE, REMOVE THE FOLLOWING TWO       IMME0056
*  INSTRUCTIONS.                                                           IMME0057
       CMPA   0,DL                                                         IMME0058
       TMI    GIVEUP        ... CORE CAN BE RELEASED                       IMME0059
*                                                                          IMME0060
*  LOOP TO FETCH REQUIRED CORE IN 2K INCREMENTS                            IMME0061
*                                                                          IMME0062
LOOP   STQ    TEMP3         SAVE THE NUMBER OF K
       CMPQ   1,DL          HOW MANY MORE K
       TZE    LAST          ... GET 1K MORE                                IMME0064
       MME    GEMORE        GET 2K MORE                                    IMME0065
       ZERO   0,2                                                          IMME0066
       TSX1   CNT           ... CORE REQUEST REFUSED                       IMME0067
       LDQ    TEMP3         UPDATE THE
       SBQ    2,DL            NUMBER OF K                                  IMME0069
       TZE    DONE          ... DONE                                       IMME0070
       TRA    LOOP          ... CONTINUE                                   IMME0072
LAST   MME    GEMORE        GET THE LAST 1K BLOCK                          IMME0073
       ZERO   0,1                                                          IMME0074
       TSX1   CNT           ... CORE REQUEST REFUSED                       IMME0075
       TRA    DONE          ... DONE                                       IMME0076
*                                                                          IMME0077
CNT    AOS    COUNT         COUNT NUMBER OF REQUESTS REFUSED               IMME0078
       LDQ    4*1000*64,DL  GO TO SLEEP FOR A WHILE                        GH041971
       MME    GEWAKE        ... ZZZ Z Z  Z   Z    Z     Z                  IMME0080
       TRA    -3,1          ... TRY AGAIN                                  IMME0081
*                                                                          IMME0082
*  SEQUENCE TO RELEASE SURPLUS CORE                                        IMME0083
```

```
*                                                                           IMME0084
GIVEUP SZN    **,DU          CHECK RELEASE FLAG
       TPL    DONE           ...POSITIVE, NO RELEASE REQUESTED
       LCQ    TEMP           GET ABSOLUTE VALUE OF NUMBER OF BLOCKS
       QLS    10             MULTIPLY BY 1024                               IMME0086
       EAA    DONE           SET RETURN ADDRESS FOR GEMREL                  IMME0087
       MME    GEMREL         ... RELEASE CORE                               IMME0088
*                                                                           IMME0089
*  PRINT STATISTICS FOR THIS CALL TO GIMME                                  IMME0090
*                                                                           IMME0091
DONE   MME    GELAPS         COMPUTE PROCESSOR TIME USED BY GIMME           IMME0092
       SSQ    TEMP2                                                         IMME0093
       MME    GETIME         COMPUTE ELAPSED TIME                           IMME0094
       SBQ    TEMP2+1                                                       IMME0095
       TPL    *+2            ...DID NOT PASS MIDNIGHT
       ADQ    =5.5296E9      ADD 24 HOURS TO CORRECT
       TOV    *+1            CONVERT TO FLOATING POINT                      IMME0096
       LDA    0,DL                                                          IMME0097
       LDE    =71825,DU                                                     IMME0098
       FNO                                                                  IMME0099
       FDV    =2.304E8       CONVERT TO HOURS FROM 64THS OF MILLISEC.       IMME0100
       FST    TEMP2+1        SAVE FOR PRINTING                              IMME0101
       LDQ    TEMP2          CONVERT PROC. TIME TO HOURS
       TOV    *+1
       LDA    0,DL
       LDE    =71825,DU
       FNO
       FDV    =2.304E8
       FST    TEMP2
       CALL   .FPRN.(FILE,FORM1)
       LDA    TEMP           PRINT NUMBER K CORE
       TSX1   .FCNV.
       LDA    COUNT          PRINT NUMBER OF REQUESTS REFUSED               IMME0103
       TSX1   .FCNV.                                                        IMME0105
       LDA    TEMP2          PRINT PROCESSOR TIME USED BY GIMME             IMME0106
       TSX1   .FCNV.                                                        IMME0107
       LDA    TEMP2+1        PRINT ELAPSED TIME                             IMME0108
       TSX1   .FCNV.                                                        IMME0109
       CALL   .FFIL.                                                        IMME0110
       STZ    COUNT          RESET FOR NEXT CALL TO GIMME                   IMME0111
       RETURN GIMME                                                         IMME0112
*                                                                           IMME0113
ERROR  CALL   FXEM(=61,MESS,=5)
```

```
RLSFLG EQU      GIVEUP
LIMITS BOOL     37
LOWLOD BOOL     24
TEMP   BSS      1                                                   IMME0115
TEMP2  BSS      2              STORAGE FOR PROCESSOR AND ELAPSED TIME IMME0116
TEMP3  BSS      1
COUNT  ZERO                                                         IMME0117
MESS   BCI      5,ARRAY NOT LOADED ABOVE PROGRAM
FORM1  BCI      9,(19H GIMME: REQUEST FOR,I4,10HK REFUSED ,I5,13H TIMES,
       BCI      8, USED ,F7.6,11H HR, PROC.,,F7.4,11H HR, ELAPS.)
       BLOCK    GIMMEB
FILE   DEC      06
       END                                                          IMME0121
$      GMAP     DECK,COMDK                           77739      011671TIME
$      INCODE   IBMF
       LBL      TIME,TIME AND DATE SUBPROGRAM                       TIME0010
       TTL      DRILL CONVERSION PROGRAM                            TIME0020
       TTLS     TIME, DATE, AND ELAPSED TIME SUBPROGRAM             TIME0030
*                                                                   TIME0040
*     CALL TIMDAT (TIME,DATE)                                       TIME0050
*                                                                   TIME0060
* WHERE  TIME = 2 CONSECUTIVE WORDS WHERE THE CURRENT TIME WILL BE  TIME0070
*               PLACED AS HH:MM:SS                                  TIME0080
*        DATE = 2 CONSECUTIVE WORDS WHERE THE CURRENT DATE WILL BE  TIME0090
*               PLACED AS MM/DD/YY                                  TIME0100
*                                                                   TIME0110
TIMDAT SAVE     0                                                   TIME0120
       MME      GETIME           GET DATE IN A AS MMDDYY AND TIME IN Q IN TIME0130
       REM                       64'THS MSEC SINCE MIDNIGHT.        TIME0140
       STQ      TIME             SAVE TIME                          TIME0150
       LRL      36               PLACE DATE IN Q                    TIME0160
       LDA      =6H              PLACE SPACES IN A                  TIME0170
       LLR      12               MOVE MM INTO A                     TIME0180
       ALS      6                                                   TIME0190
       ORA      =3H00/,DL        INSERT SLASH                       TIME0200
       LLR      12               MOVE DD INTO A                     TIME0210
       ALS      6                                                   TIME0220
       ORA      =3H00/,DL        INSERT SLASH                       TIME0230
       LDX0     3,1              LOAD ADDRESS OF DATE               TIME0240
       STA      0,0              STORE FIRST                        TIME0250
       STQ      1,0              AND SECOND WORD                    TIME0260
       LDQ      TIME             PUT TIME IN Q                      TIME0270
       QRS      6                CONVERT TO MSEC                    TIME0280
```

```
        DIV     1000,DL         CONVERT TO SEC
        DIV     10,DL
        STA     TIME            STORE UNIT SECONDS
        DIV     6,DL
        STA     TIME+1          STORE TEN'S OF SECONDS
        DIV     10,DL
        STA     TIME+2          STORE UNIT MINUTES
        DIV     6,DL
        STA     TIME+3          STORE TEN'S OF MINUTES
        DIV     10,DL           UNIT HRS TO A, 10'S OF HRS IN Q
        ALS     30
        LLR     6               ASSEMBLE HH IN RIGHT OF Q
        QLS     6
        ORQ     =015,DL         INSERT COLON
        QLS     6
        ORQ     TIME+3
        QLS     6
        ORQ     TIME+2          INSERT MM
        QLS     6
        ORQ     =015,DL         INSERT COLON
        LDA     =6H             PLACE SPACES IN A
        LLS     6
        ORQ     TIME+1
        LLS     6
        ORQ     TIME            INSERT SS
        LLR     24
        LDX0    2,1             LOAD ADDRESS OF TIME
        STA     0,0             STORE FIRST AND
        STQ     1,0             SECOND WORD
        RETURN  TIMDAT
        EJECT
*
*       CALL ELTIME(TIMEL)
*
* WHERE  TIMEL = LOCATION WHERE ELAPSED TIME IN MSEC. IS PLACED.
*
ELTIME  SAVE
        MME     GELAPS          GET ELAPSED TIME IN Q IN 64'THS MSEC.
        QRS     6               CONVERT TO MSEC
        STQ     2,1*            STORE IN TIMEL
        RETURN  ELTIME
TIME    BSS     4
        END
```

```
$       EXECUTE
$       FFILE   07
$       FILE    08,,1R
$       LIMITS  200,17K,,20K
TITLE THERMAL TEST CASE
SIZE        50
HEAT         1     5
$       ENDJOB
```